FINAL ADMINSTRATIVE REPORT

Title: GSRP/David Marshall: Fully Automated Cartesian Grid CFD Application for MDO in High Speed Flows

Sponsor: NASA Ames Research Center

University: Georgia Institute of Technology

Project Director: Stephen M. Ruffin

GT P/S Project No.: 1606R84

GT Old Project No: E-16-R84

Contract No.: NGT 2-52266

Date: November 14, 2003

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Alphanumeric

$\boldsymbol{0}$	Zero vector
A	Area
a	Speed of sound
CS	Control surface of integration
CV	Control volume of integration
C_p	Specific heat at constant pressure
C_v	Specific heat at constant volume
e_t	Total energy per unit mass of control volume
$e_{internal}$	Internal energy of control volume
$\boldsymbol{g}_{\oplus}$	Acceleration due to earth's gravity
$\boldsymbol{h}$	Distance vector from common datum for potential energy calculation
$\boldsymbol{n}$	Surface normal vector
$\bar{\mathrm{n}}$	Surface normal vector
p	Pressure
q	Scalar heat flux value
$\boldsymbol{q}$	Heat flux vector
$\boldsymbol{r}$	Radius from rotation point

u,v,w	Velocities in the x-, y-, and z-directions
$\boldsymbol{v}$	Velocity vector
$\bar{v}$	Velocity vector
$\bar{v}_n$	Component of velocity normal to surface
$\bar{v}_t$	Component of velocity tangent to surface
x,y,z	Physical coordinate directions
E_n	Parallelization efficiency for n processors
$\boldsymbol{F}$	Function vector of system of 1st order ODEs
F_b	Body force acting on control volume
$\boldsymbol{i}$	Unit vector in the x-direction
$\boldsymbol{j}$	Unit vector in the y-direction
$\boldsymbol{k}$	Unit vector in the z-direction
$\boldsymbol{K}$	Right eigenvectors of matrix
N	Number of control volumes in grid
R	Gas constant
SU_n	Parallelization Speed-up for n processors
T	Temperature
U	Speed
$\boldsymbol{U}$	Velocity vector
$\boldsymbol{U}$	State vector of conserved variables
$\boldsymbol{U}$	Solution vector of system of 1st order ODEs

W State vector of primitive variables

Caligraphic

$\mathscr{F}_I$ Inviscid flux

$\mathscr{F}_V$ Viscous flux

$\mathscr{H}$ One-dimensional hyperspace

$\mathscr{O}$ Order of magnitude

$\mathscr{R}^d$ d dimensional space

$\mathscr{V}$ Volume

Greek and other

α Wave speed in Roe's Approximate Riemann Solver

γ Ratio of specific heats

ε Tolerance

κ Thermal conductivity

λ Eigenvalue of a matrix

μ Dynamic viscosity

ν Kinematic viscosity

ρ Density

τ Shear stress

$[\tau]$ Shear stress tensor

τ_x Component of shear stress tensor acting in x-direction (first row of shear stress tensor)

τ_y Component of shear stress tensor acting in y-direction (second row of shear stress tensor)

τ_z Component of shear stress tensor acting in z-direction (third row of shear stress tensor)

τ_{xx} Shear stress acting on x-face in x-direction

τ_{xy} Shear stress acting on x-face in y-direction

τ_{xz} Shear stress acting on x-face in z-direction

τ_{yx} Shear stress acting on y-face in x-direction

τ_{yy} Shear stress acting on y-face in y-direction

τ_{yz} Shear stress acting on y-face in z-direction

τ_{zx} Shear stress acting on z-face in x-direction

τ_{zy} Shear stress acting on z-face in y-direction

τ_{zz} Shear stress acting on z-face in z-direction

Ω Angular velocity vector

∞ Shorthand for $\lim_{s \to \infty} s$

$\triangle_{xyz}$ Triangle with vertices x, y and z

& Bitwise "and" operator

| Bitwise "or" operator

Special Constants

Ec Eckert number

Fr Froude number

Pr Prandtl number

Re Reynolds number

Superscripts

$*$ Non-dimensionalized quantity

n n^{th} derivative of single value function

Subscripts

max maximum item

n n^{th} item

w,$wall$ wall

∞ infinity, freestream values

Abbreviations

AMR Adaptive Mesh Refinement

API Application Programming Interface

CFD Computational Fluid Dynamics

CPU Central Processing Unit

FAS	Full Approximation Storage
IPC	Interprocess Communication
MIMD	Multiple Instruction Multiple Data
MUSCL	Monotone Upstream-centered Scheme for Conservation Laws
MPI	Message Passing Interface
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PVM	Parallel Virtual Machine
SFC	Space-Filling Curves
SIMD	Single Instruction Multiple Data
const.	Constant

Operators

$\frac{\partial}{\partial x}$	Partial derivative with respect to x
$\frac{\partial^n}{\partial x^n}$	n^{th} partial derivative with respect to x
$\frac{d}{dx}$	Derivative with respect to x
$\frac{d^n}{dx^n}$	n^{th} derivative with respect to x
∇	Gradient

Miscellaneous Symbols

$\approx$	approximately equal

$|_{\bullet}$ Evaluated at $\bullet$

$\gg$ Much greater than

$\ll$ Much less than

SUMMARY

With the renewed interest in Cartesian gridding methodologies for the ease and speed of gridding complex geometries in addition to the simplicity of the control volumes used in the computations, it has become important to investigate ways of extending the existing Cartesian grid solver functionalities. This includes developing methods of modeling the viscous effects in order to utilize Cartesian grids solvers for accurate drag predictions and addressing the issues related to the distributed memory parallelization of Cartesian solvers.

This research presents advances in two areas of interest in Cartesian grid solvers, viscous effects modeling and MPI parallelization. The development of viscous effects modeling using solely Cartesian grids has been hampered by the widely varying control volume sizes associated with the mesh refinement and the cut cells associated with the solid surface. This problem is being addressed by using physically based modeling techniques to update the state vectors of the cut cells and removing them from the finite volume integration scheme. This work is performed on a new Cartesian grid solver, NASCART-GT, with modifications to its cut cell functionality. The development of MPI parallelization addresses issues associated with utilizing Cartesian solvers on distributed memory parallel environments. This work is performed on an existing Cartesian grid solver, CART3D, with modifications to its parallelization methodology.

CHAPTER I

INTRODUCTION

Computational Fluid Dynamics (CFD) researchers have always had to strike a balance between the accuracy and fidelity of their model with the efficiency and availability of the computational hardware. Early on many sacrifices to the accuracy and fidelity of the model were needed in order to accommodate the available computational hardware. Now techniques and more powerful computational hardware exist that yield more accurate numerical simulations in complex flow fields. One of the early schemes that has gained renewed interest is the use of Cartesian grids. A benefit of using Cartesian grids is that the number of terms needed in the solution procedure for the governing equations is greatly reduced compared to more elaborate gridding techniques since the edges of the control volumes are coordinate aligned and thus no need for the more complex contravariant velocity formulations. Also, the ability to easily create grids for very complicated geometries makes Cartesian grids an attractive approach to CFD. The drawback is the complexity associated with the computational cells that intersect the geometries as well as the inability of the traditional Cartesian grid formulations to model viscous flows. The present chapter presents an overview of Cartesian grid methods, Navier-Stokes techniques and parallelization approaches, and concludes with the motivation for the present work.

Cartesian Grid Origins

Cartesian grids have been utilized in solving a variety of CFD problems from potential flows [13, 135, 184] to the Euler equations [23, 35, 36, 84, 105, 189] to the Navier-Stokes equations [38, 39, 49, 59, 79, 180, 178]. Cartesian grids consist of a collection of non-overlapping, connected control volumes with coordinate aligned edges. Thus, the edge (or face in three dimensions) normals for all complete cells are aligned with one of the coordinate directions. Figure 1 shows a typical two-dimensional Cartesian grid around a curved surface.

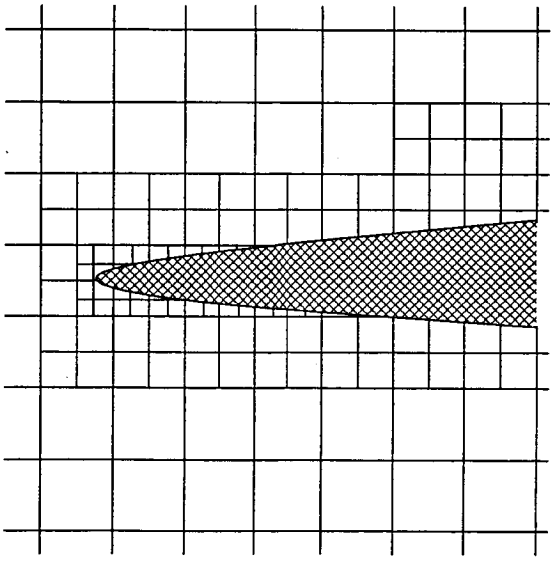
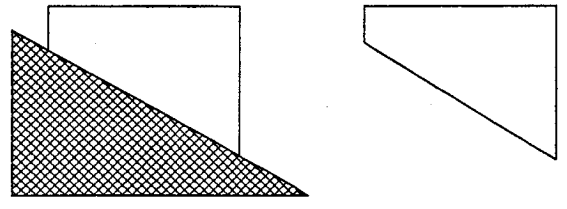


Figure 1: Example Cartesian Grid Near Curved Surface

Cartesian gridding techniques have become the focus of recent research due to their ability to easily handle complex geometries in the grid generation phase, the ease with which higher order schemes can be applied and the natural connection between the grid

refinement techniques and multigrid acceleration schemes [105]. The difficulties in using Cartesian grids arise from the fact that the control volumes adjacent to the surfaces are not usually aligned with the surfaces and thus special techniques need to be employed to handle the non-Cartesian (cut or split) cells in these regions.

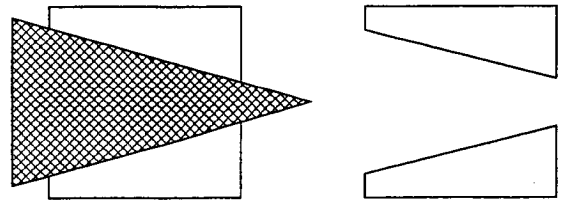
Cut cells are created when the intersection of the Cartesian cell and the solid surface results in one computational volume with only a fraction of the original volume and possibly non-Cartesian aligned edges, see Figure 2. Split cells are created when the intersection of the Cartesian cell and the solid surface results in two or more computational volumes which might have non-Cartesian aligned edges, see Figure 3.



Solid surface overlayed Cartesian Cell

Resulting Cut Cell

Figure 2: Example of Cut Cell Creation



Solid surface overlayed Cartesian Cell

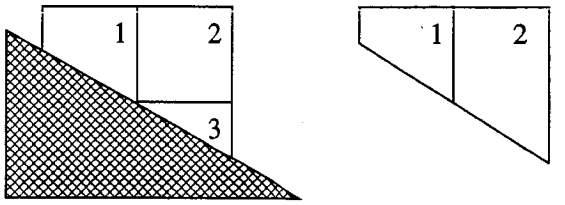
Resulting Split Cells

Figure 3: Example of Split Cell Creation

The original use of Cartesian grids involved solving the two-dimension full potential equation by Purvis and Burkhalter [135], followed shortly afterwards by Wedan and South [184], in which a non body-oriented structured grid was created on which the full potential equation was solved. Their solution strategy was to use finite volume techniques in order to more easily handle the computational cells that were intersected by the solid surface. Additionally, they used linear approximations in the cut cells for the reconstruction of the wall boundary conditions which provided a simple algorithm for implementation and preserved the structure of their coefficient matrix during the solution iteration so that no extra computational costs were incurred for the cut cells. However, this did not preserve the actual body curvature and also only provided a linear approximation to the actual surface lengths and area for the cut cells, and thus could not exactly model curved surfaces. Also, little mention was made of any attempts at cell refinement to more accurately capture the surface geometry and flow features.

Later, Clarke et al. [36] used Cartesian grids to solve the two-dimension Euler equations (again on non grid-aligned surfaces). They attempted to more accurately model the solid surface boundary conditions by utilizing the local surface curvature in reconstructing the wall boundary conditions. They also provided more accurate modeling of the cut cell lengths and areas by using the actual surface geometry in their calculations and not linear approximations. Additionally, they noted that clustering was needed in certain critical regions in order to produce accurate results, and this was achieved by clustering entire grid lines. Cut cells that were too small (less than 50% of the original cell size) were merged

with neighbor cells in order to avoid time stepping problems associated with very small computational cells. Gaffney and Hassan [60] extended this research to three dimensions. Figure 4 demonstrates the case of cell merging. In this case the surface cuts through a collection of cells, numbered 1–3. Cell 1 turns into a cut cell (numbered 1 in the resulting merged cells) while cells 2 and 3 are merged together into the cell numbered 2 since cell 3 is too small after the cut.



Solid surface overlayed Cartesian Cells

Resulting Merged Cells

Figure 4: Example of Merge Cell Creation

Adaptive Mesh Refinement

Berger and LeVeque [23] addressed several deficiencies that existed in the established uniform grid methodologies. First, they applied the concept of Adaptive Mesh Refinement [24] (AMR) in order to improve the accuracy in critical regions without adversely affecting the efficiency of the numerical integration scheme. The use of AMR effectively allowed the clustering of blocks of computational grids as the solution process evolved only in the region that they were needed (and not clustering entire grid lines), by using Richardson-type

extrapolation error estimates to identify regions of large errors and adding grid blocks in those regions. An example of AMR is Figure 5 which represents a simple adapted grid for a supersonic wedge flow with four levels of adaption. As can be seen in the figure, there are more control volumes where gradients are to be expected, specifically along the surface to capture the geometry and along the oblique shock. In regions with small gradients, there is a lower density of control volumes. Also notice that in this figure there is at most a 2:1 ratio at the refinement interface, which is typical of most AMR schemes, in order to promote stability in the numerical schemes.

One problem with Berger and LeVeque's original implementation of AMR on Cartesian grids was the problem of state variable conservation during the AMR stages. They carefully constructed conservative schemes for the inter-grid transfer to address the problem. They also used the idea of wave propagation and directional differencing [89] in order to increase the stability near the small boundary cells. This helped keep the CFL of the boundary cells reasonably close to the CFL of the flow cells and allowed larger time steps to be taken with the solver remaining stable.

Several researchers have extended Berger and LeVeque's research into areas such as multigrid Cartesian grids [55, 56], higher accuracy flow solvers using more sophisticated flux approximations [45, 46], time-accurate unsteady flows [35], and a front tracking AMR scheme [126, 127] that attempted to track the discontinuities (such as shocks) as the solution evolved in order to provide more accuracy in the refined mesh calculations. Quirk had developed an AMR based software architecture called AMRita [136, 137], a software

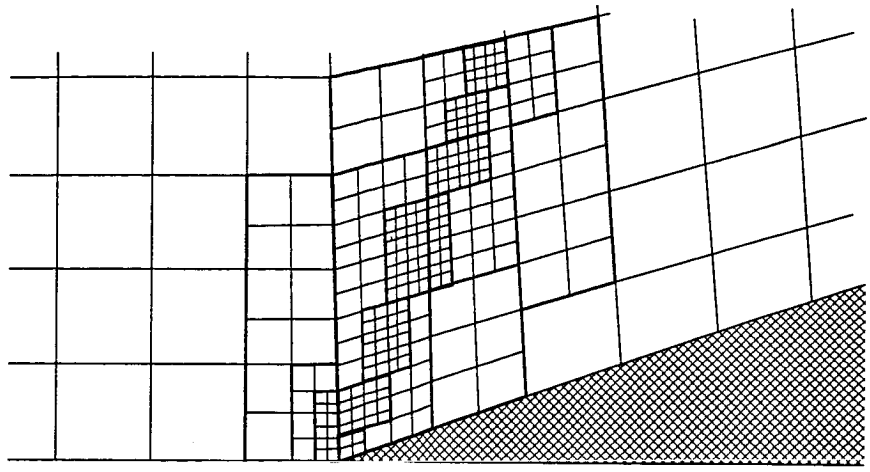


Figure 5: Example Adaptive Grid for Supersonic Wedge Flow

system for automating numerical investigations, that attempts to abstract out much of the tedium associated with developing and testing CFD software.

Advanced Geometry Modeling

Melton et al. [105] developed techniques for handling more complex surface geometries using Cartesian gridding techniques. They extracted the surface geometry from CAD/CAM compatible geometry definitions and used higher-order surface modeling techniques to determine the cut cell geometries. This provided more accurate solid surface reconstructions which resulted in more accurate solid surface boundary conditions. They also addressed surface refinement issues that arise from the intersections of arbitrary geometries and the computational cells. When an arbitrary geometric surface (or set of surfaces) intersected the

computational volume, multiple intersections could occur within one cell or multiple independent computational regions could be created. They developed an automated technique that detected these cases and refined these regions with little or no user input. The result of this effort was an application that could extract surface geometries from CAD/CAM models, generate the computational grids, and solve the fluid dynamics equations. Extensions of this effort have been done by Melton et al. [104] with improvements to the grid generation algorithms as well as the geometry refinement schemes and the geometry representations.

As an extension to the work performed by Melton and his colleagues, Aftosmis et al. [3, 4, 22] developed a Cartesian grid application (CART3D) that provided a number of improvements over the original work. Their major focus was on providing accurate and robust resolution of the cut cell geometries and high performance improvements to the solution methodology. Their work on the cut cell geometries dealt with providing a systematic way of addressing and handling the variety of cut cell types that could occur when a surface with an arbitrary number of facets intersects a computational cell. Along with automatic handling of cut cells, split cells and merged cells, they also applied a sub-cell resolution procedure to the solid surfaces of the cut and split cells in order to improve the accuracy of the surface modeling. This entailed generating a normal for each surface patch from the original geometry definition that intersected the control volume. In addition, a surface normal agglomeration technique was developed for the cut and split cells could be used in order to improve the computational efficiency of the code without sacrificing significant accuracy. A comprehensive description of this research can be found in reference [2].

In an effort to handle more complex geometries in computational aeroacoustics configurations, Kurbatskii and Tam [84] developed a method of treating solid surfaces in high-order numerical schemes without loosing the acoustic wave speed accuracy associated with the less dispersive and dissipative high-order schemes in computational aeroacoustics. Their research utilized a uniform two-dimensional mesh and solid boundary ghost cells with coarseness limitations imposed by the body surface curvature that ensured simple cut cell geometries. They used the body curvature to develop accurate body pressure values that could be applied to linear surface approximations and still retain the desired accuracy. In order to achieve this accuracy, a linear system of equations on the order of the number of surface cells needed to be solved in order to generate the required ghost cell pressures which could cause a negative impact on the overall performance of the scheme.

Another research direction that evolved from the Cartesian grid research was the study of unsteady flows, especially about moving bodies. Chiang et al. [35] were one of the first researchers to study the unsteady Euler equations on Cartesian grids and provided an analysis of two techniques to adequately capture the unsteady effects: (1) small grid cells and (2) high-order accurate schemes. Bayyuk et al. [19] addressed the issue of moving and deforming bodies by defining the motion of the body through the pre-existing Cartesian grid in two dimensions with discussions on the extension to three dimensions, without results, by Lahur and Nakamura [86]. As the body moved, mesh refinement occurred in order to capture the surface geometry in its new location. Cell merging occurred when the body cut a computational cell into a volume that fell below some specified threshold, as

well as when cells were just being exposed due to the body motion. One drawback to this procedure was that there was a limit placed on the time step that depended on the smallest cell size and the body motion such that the body could not sweep through an entire volume in one time step. Yang et al. [189] developed a similar solver from an existing stationary body solver [188] and also encountered the time-step limitation due to the body sweeping over an entire cell.

One final approach to solving the moving body problem was presented by Murman et al. [115] in which an arbitrarily large time step is allowed by using a space-time conservation approach [88, 194] to account for the effects of the body sweeping entirely through a control volume in one time step for a three-dimensional configuration. This approach exactly satisfies the geometric conservation laws for most cells in the flow at each time step with some cells only approximately satisfying the geometric conservation laws.

Navier-Stokes Modeling

Numerical solution of the Navier-Stokes equations has been the focus of many researchers throughout the history of Computational Fluid Dynamics (CFD), and a number of different approaches have been utilized. Generally, the attempts fall into three categories: (1) solutions of the full Navier-Stokes equations over the entire computational domain, (2) solutions of approximations to the Navier-Stokes equations over the entire computational domain and (3) solutions of approximations to the Navier-Stokes equations in a subdomain of the entire computational domain.

Prior to the 1980's, solution of the full Navier-Stokes equations over the entire computational domain was normally considered outside of the available computational resources [134, 152]. Thus early research into generating computational solutions to the Navier-Stokes equations primarily focused on techniques (2) and (3). These methods will be reviewed on page 11 and page 15 respectively, followed by a discussion of fully resolving the viscous terms in the Navier-Stokes equations for Cartesian grids on page 17.

Navier-Stokes Approximations

There are two approximation techniques of interest to Cartesian solutions to the Navier-Stokes equations. The first is the thin-layer Navier-Stokes approximation that has only limited use in pure Cartesian formulations, but can be useful in the chimera or hybrid schemes discussed later. The second is the vorticity confinement technique that uses an extra force term in the momentum equations to prevent the numerical dissipation of vortices and model the vortical regions created by the boundary layers in the flow.

Thin-Layer Navier-Stokes Approximation

The thin-layer approximation to the Navier-Stokes equations was developed from a dimensional analysis of the governing equations for high Reynolds number flows. By eliminating terms that produced higher order effects, sufficiently accurate solutions to the Navier-Stokes equations could be developed in a reasonable amount of time on the computational hardware available. Ultimately, this effort resulted in a solution that resolved the viscous stresses normal to the body (or bodies) in a thin region while the other directions

used the inviscid fluxes only.

Steger [152] developed the thin-layer Navier-Stokes approximations as a means of obtaining solutions to three-dimensional flows with high Reynolds numbers, while at the same time Baldwin and Lomax [16], as well as Pulliam and Steger [134], demonstrated similar ideas for high Reynolds number turbulent flows. The general reasoning behind this scheme was that the current computational power and memory requirements would not allow adequate grid resolutions in all coordinate directions, so a dimensional analysis was performed on the full Navier-Stokes equations to try to eliminate terms. This analysis showed that in order to adequately resolve the viscous terms along the body, $\mathcal{O}\left(\frac{1}{\sqrt{Re}}\right)$ grid spacing would be required in each direction. This level of clustering would require a prohibitively large amount of CPU time and memory. In high Reynolds number viscous flows, the viscous terms were dominated by the wall normal derivatives [186], thus the thin-layer Navier-Stokes approximations neglected all viscous terms that were not in the surface normal direction. Then, by generating a body-oriented structured grid, the thin-layer terms could easily be retained by eliminating the terms in the coordinate direction(s) along the body surface in the viscous flux calculations. This resulted in a thin, viscous boundary layer around the solid surfaces that adequately resolved much of the viscous effects in the flow, including separation points, while obtaining results in a reasonable amount of time from the computational hardware available.

This research resulted in the computational packages ARC2D and ARC3D [132] that have been in use for many years [133, 154] and have been the basis of other efforts, see

references [120] and [149] for examples. Additionally, modeling such effects as thermal boundary layers and isothermal walls were not explicitly precluded by the thin-layer approximations, as long as these effects were dominated in the body normal direction (as they typically were for the cases being studied at the time), however capturing flow phenomena such as leading edge effects and separated regions was beyond the capacity of this approach due to the high streamwise viscous stresses present.

Vorticity Confinement

The vorticity confinement technique has its origins in the front tracking schemes, such as shock capturing methods, that attempt to track a sharp discontinuity by using Lagrangian elements in a flow field of an Eulerian based solver. The vorticity confinement approach, developed by Steinhoff and others[51, 58, 73, 112, 155, 156, 185], uses the fact that the vortical regions, from shed vortices and the boundary layer, in high Reynolds number flows are very small.

For the shed vortices, a forcing function in the direction normal to the vorticity is applied to the momentum equations in these regions to convect the vorticity back to the centroid of the cell. This technique has been found to be quite useful for capturing shed vortices as they travel long distances through inviscid flow fields without distorting the original vortex strength direction.

For the boundary layer regions, a forcing function related to the distance of the cell to the wall is used to advect the vorticity back to the surface. In order to enforce the no-slip boundary condition, the domain inside the body and on the surface is forced to have zero

velocity.

The current implementations of vorticity confinement have been limited to uniform Cartesian grids and body conforming grids. Attempts to extend this technique into more irregular mesh topologies have had limited success because of the dependency of the confinement parameter on the grid cell size. Without varying the confinement parameter, Murayama and Nakahashi [114] found premature vortex bursting on a delta wing for an unstructured grid formulation. Löhner and Yang [93] have recently attempted to address the confinement parameter limitation with a dimensional analysis of the confinement parameter and have demonstrated some favorable results.

This technique allows the use of much coarser grids to model high Reynolds number flow fields that have compact vortices. However, it does not capture any of the details of the interior of the vortical regions as it only models these regions as thin lines. Further, care must be taken in setting the confinement parameter in order to avoid the problems discussed by Dietz et al. [51] where the vortical regions become unphysical. There is concern [93] that the vorticity confinement, which is introduced as a force term in the momentum equations, might alter the local axial and tangential momentum. However, this is a promising approach and warrants further study.

Viscous/Inviscid Coupling

The other main technique used to provide approximate solutions to the Navier-Stokes equations was a technique of coupling an inviscid solver for the majority of the computational domain with a solver that captured the viscous terms for the regions near the solid surfaces (or other high viscous regions). The justifications for this technique were similar to those presented for the thin-layer Navier-Stokes solutions, i.e. high Reynolds number flows confine the viscous effects to small regions where high gradients occur (such as boundary layers and shear layers).

Carter [30, 31] and Vatsa and Carter [168], and later Van Dalsem and Steger [162] as well as Kaups and Cebeci [81], were some of the first researchers to develop the viscous/inviscid coupling techniques for CFD applications. Their solution procedure started with the development of boundary layer equations for their solver configurations using standard dimensional analysis techniques which resulted in the familiar boundary layer equations [186]. The solution procedures for the boundary layer equations mainly focused on inverse boundary layer algorithms in order to model small separation regions that the direct boundary layer algorithms cannot handle due to the singularity at the separation point [7]. These equations were typically solved on body-oriented structured grids that captured the entire boundary layer. For the inviscid calculations, early research focused on solving the potential equations using body-oriented structured grids that overlay the boundary layer grids. Later efforts focused on using the Euler equations as the inviscid model [138, 34] as well as solving the Euler equations on unstructured grids [131].

Modeling the viscous/inviscid interaction was done by using the transpiration velocity concept [162] or by using the boundary layer displacement approach [34]. The transpiration velocity concept used the velocity components as a means of vorticity transport from the viscous regions to the inviscid regions. This method imposed a requirement on the inviscid mesh that it be fine enough to accurately resolve the vorticity near the surface [154]. The velocity differences were then applied to the inviscid velocities which resulted in a blowing-type surface boundary condition [33]. The boundary layer displacement approach used the inviscid solution to calculate the boundary layer thicknesses and then modified the solid body geometry in the next step of the inviscid solver to include the calculated boundary layer thicknesses. It is worth noting that neither the transpiration velocity approach nor the boundary layer displacement approach paid any significant attention to the thermal boundary layer effects as this research was mainly focused on the subsonic to transonic regime.

Drela and Giles [53] extended the viscous/inviscid solution concept by developing a formulation to handle low Reynolds number flows. Additionally, they strongly coupled the two solution regimes by solving the entire nonlinear equation set via a global Newton-Raphson iterative method. The resulting code was called ISES (and its successor MISES [193]) and has been used extensively in aerodynamic design studies such as in reference [147].

Navier-Stokes and Cartesian Grids

While the majority of research into Cartesian grids has focused on solving the Euler equations in two- and three-dimensions, there has been some notable efforts into the utilization of Cartesian grids to solve the Navier-Stokes equations. These efforts have focused on either solving the full Navier-Stokes equations using either the immersed boundary methods [64, 110, 128], volume-of-fluid methods [12, 67, 70], reconstruction based schemes [95, 190] or cut cell based techniques [38, 59, 178] or coupling body-fitted grid solutions of the Navier-Stokes equations with a Cartesian background grid [13, 21, 48, 55, 79]. The grid coupling technique has its foundations in the idea of the viscous/inviscid coupling discussed on page 15.

Note that the other early approach to the Navier-Stokes equations was the thin-layer approximations discussed on page 11 and has found little use in Cartesian grids because the thin-layer Navier-Stokes approximations relied on the grid being body oriented. Cartesian grids do not, in general, provide grids that are body aligned, however some work has been performed applying the thin-layer techniques to Cartesian grids [59]. Hybrid methods do exist which couple a body oriented grid solving the thin-layer Navier-Stokes equations with a background Cartesian grid [103].

Immersed Boundary Methods

The immersed boundary method was originally developed by Peskin [128, 129] for heart valve modeling using the Navier-Stokes equations in two dimensions. The heart

valves were modeled as flexible surfaces that can propagate with the flow, subject to certain limitations such as hinge points or rigid regions on the surfaces. Instead of remeshing the computational domain as the surface is propagated, the cells that contain the surface have a body force added to their momentum equations that represents the reactive force that the body is applying to the fluid in response to the fluid surface pressure and shear stress.

Goldstein et al. [64] applied Peskin's work to incompressible, solid body flows using a force feedback approach. In this formulation, the surface force takes the form of a feedback loop function that acts on the surface cell to bring the surface velocity to zero by adjusting the applied forces appropriately. This approach requires an extremely small time step (CFL around 10^{-3}) in order for it to remain stable.

The small time step limitation of Goldstein et al. was addressed in the work by Mohd-Yusof [110, 111]. Here, the incompressible Navier-Stokes equations are solved using a pseudo-spectral method. The applied body force is developed by utilizing the time-discretized Navier-Stokes equations on the surface. In order to generate a smooth no-slip boundary condition, forces are also applied to the cells adjacent to the surface.

In order to more accurately determine the appropriate surface forces to add to the momentum equations, Fadlun et al. [57] developed a second-order boundary interpolation scheme for three-dimensional incompressible flows by using linear interpolation to reconstruct the state information at the surface. This approach resulted in the use of larger time steps (CFL around 1.5) and better accuracy at the surface. Further advances by Lai and Peskin [87] developed second-order methods for moving membranes. Additionally, Kim et

al. [82] developed a second-order method with both momentum and mass sources in order to improve the overall accuracy of their results.

While these schemes handle the Navier-Stokes equations on Cartesian grids, they all suffer from numerical stability problems that typically require numerical diffusion. Also, the surface is not sharply resolved, and is typically smeared between 2 or 3 cells. This can cause problems when flow details are needed near the surface.

Volume of Fluid Methods

Another approach to solving the Navier-Stokes equations on Cartesian grids is the volume of fluid method. In this method, a scalar transport equation is solved in addition to the Navier-Stokes equations. The scalar is a value between 0 and 1 that represents the volume fraction that the fluid (or gas) occupies in that cell. The typical use of this scheme is free-surface flows, where the scalar represents the amount of the cell that the fluid occupies, and interfacial flows, where the scalar represents the volume fraction that a species occupies in the cell.

Hirt and Nichols [70] originally developed this method as part of an incompressible free-surface Navier-Stokes solver. In order to retain the incompressible invariance in the transport equation, strict mass conservation was required of the numerical solver. They also used a first order accurate surface reconstruction technique which causes problems resolving the interface boundaries.

Ashgriz and Poo [12] were one of the first researchers to develop a piecewise linear interface construction technique to better resolve the interface boundaries. This is the most

popular technique currently in use for interface reconstruction. Almgren et al. [6] used the volume of fluid technique, coupled with a finite volume solver, to model the solid surface in incompressible viscous flows. Henderson et al. [67] and later Miller and Puckett [108] have also extended the volume of fluid technique to compressible flows.

The volume of fluid schemes typically work well when the interface curvature is small with respect to the surface modeling. Otherwise, artificial discontinuities can develop as well as the inability to resolve the small scale features at the interfaces. Additionally, without accurate propagation of the scalar transport equation and sophisticated schemes to resolve the interface boundaries, artificial mixing can occur. Finally, problems can develop if there is no limiter placed on the scalar transport propagation to strictly enforce the scalar values in the range of 0 to 1. Scardovelli and Zaleski [145] provide a nice review of the application of the volume of fluid technique to free-surface and interfacial flows.

Reconstruction Schemes

Another class of schemes used to solve the Navier-Stokes equations on Cartesian grids are the reconstruction based schemes. These have been proposed by Ye et al.[190, 191] and Majumdar et al.[95]. These schemes are all based around the idea of interpolating the state information to the nodes in the computational domain around the surface.

Ye et al. [190, 191] have developed a two-dimensional incompressible Navier-Stokes equation solver. The solver use the cell merging technique to eliminate any surface cells that are smaller than 50% of their full size. Then, the state information for the faces of the new cell are found by utilizing a linear-quadratic two-dimensional interpolation from

the surrounding cells. This technique results in a slow convergence of the pressure Poisson equation and requires acceleration techniques. This technique has been extended to moving boundaries by Udaykumar et al. [161].

Majumdar et al.[95] have developed two-dimensional, turbulent Reynolds Averaged Navier-Stokes solver on uniform Cartesian grids. This solver uses interpolation polynomials in one- and two-dimensions to reconstruct the state of the cells that are inside the body. Thus, the solution process is performed over uniform cells at the surface. The interpolation process can cause numerical instabilities due to the negative coefficients that can arise with certain interpolation polynomials.

Cut Cell Based Methods

Frymier et al. [59] developed the first work in the application of the full Navier-Stokes equations on Cartesian grids using the cut cell approach. Their work was limited to two dimensions and laminar flows. The solution procedure was a straight-forward finite-volume approach with the Cartesian grids clustered using grid line clustering and not AMR. Their results demonstrated strong dependencies on the smoothness of the surface grid where non-smooth surface grids produced non-smooth skin-friction and surface pressure values.

A large number of standard viscous flux formulations for cut cell based schemes were analyzed by Coirier [38, 39] and Coirier and Powell [40, 41] to ascertain their accuracy and positivity characteristics. These viscous flux formulations fell into two categories: (1)

Green-Gauss reconstructions where the divergence theorem was applied to cells neighboring the face that the flux was being calculated to build the integration path and (2) polynomial based reconstructions that used a Lagrange polynomial and a set of support cells to interpolate the state variables where they were needed with the polynomial being differentiated to obtain the needed gradients. This research focused on the accuracy of the various formulations via a standard Taylor series approximation analysis and on the positivity of the formulations. The positivity is a measure of how well the discretization satisfies the local maximum principle that holds for all homogeneous, second order partial differential equations (PDEs). The local maximum principle simply states that the solution to a homogeneous, second order PDE at one point is bounded by the values of its neighbors. It is a statement of the diffusive nature of second order PDEs, and thus it is a necessary requirement for any discretization of a homogeneous, second order PDE.

The results of this effort were that all of the schemes demonstrated (to some degree) a competition between the accuracy of the scheme and the viscous stencil positivity for non-uniform cells, i.e. any attempt to improve the accuracy/positivity adversely effected the resulting positivity/accuracy. Thus, in order to achieve a higher order of accuracy, a scheme must be used that does a poor job of preserving the positivity, and vice versa. In fact, some of the schemes that were analyzed actually grid divergent, demonstrating a truncation error of $\mathcal{O}\left(\frac{1}{h}\right)$.

The resulting numerical analysis was performed for low to moderate Reynolds number

flows using a diamond-path Green-Gauss reconstruction stencil, due to its favorable positivity characteristics, and a quadratic polynomial interpolation scheme, due to its guaranteed consistency characteristics. Cases where the surface was predominantly aligned with the coordinate directions showed excellent agreement with theoretical values, but when the body was not aligned with the coordinate directions (thus, the surface had cut cells of varying volume fractions of the uncut cells) large oscillations occurred in the results due to the sensitivity of the viscous stencil to the grid smoothness (for both cut cells and coarse/fine cell interfaces). This explains the non-smooth skin friction and surface pressure values in the Frymier et al. results mentioned on page 21. Another impediment to utilizing this scheme for high Reynolds number flows was the large number of control volumes needed to adequately resolve the viscous regions. Even with AMR this became prohibitively large for even moderately complex geometries [178].

In addition to the viscous flux formulation results, AMR was applied to Coirier's solution strategies with a positive effect, but without fully eliminating the viscous stencil sensitivity on the cut cell smoothness. Another approach that was discussed was the use of embedded, body oriented grids to capture the boundary layers, but no numerical results were given. This topic of embedded body oriented grids will be discuss further on page 24.

Delanaye et al. [49] proposed a fix to the viscous stencil positivity problem by using a modified diamond-path Green-Gauss reconstruction stencil that adjusts the shape of the stencil to a more uniform shape. The state information at these points is then calculated by using a linearity preserving, pseudo-Laplacian interpolation algorithm by Holmes and

Connell [71]. While this technique was applied to a hybrid grid (a discussion of this type of gridding to follow on page 28) in two-dimensions, this scheme appears to be applicable to three-dimensional, pure Cartesian meshes.

Wang and Chen [178] developed a Cartesian grid approach to the Navier-Stokes equations that attempted to capitalize on the anisotropic nature of the viscous effects by creating anisotropic cells that can be refined in the direction(s) that the viscous effects were most dominant. This technique worked well when the direction of the dominant viscous stresses were aligned with the coordinate directions as in a flat-plate, thin wing, or similarly shaped body where the majority of its surfaces were coordinate aligned. Effective use of anisotropic refinement further required that the dominant flow direction must be aligned with a coordinate direction (and preferably in the same coordinate direction as the body). While this effort attempted to solve the problem of having a large number of computational cells, its effectiveness was limited to a small set of general configurations due to the need for favorable flow and body geometry configurations.

Chimera Grid Schemes

The use of a collection of grids to cover the computational domain is known as chimera gridding. Typically, a body-oriented structured grid is used around each component of the solid surfaces. Each of these structured grids are then overlayed onto a background Cartesian mesh. Figure 6 shows an example of a two-dimensional chimera grid collection around a simple curved surface. Notice that there is no simple mapping of cells in the body oriented grid and the background Cartesian grid. This feature is one of the drawbacks to

chimera gridding schemes, but it is only a performance penalty when the grid needs to be generated during initialization and after any AMR processes.

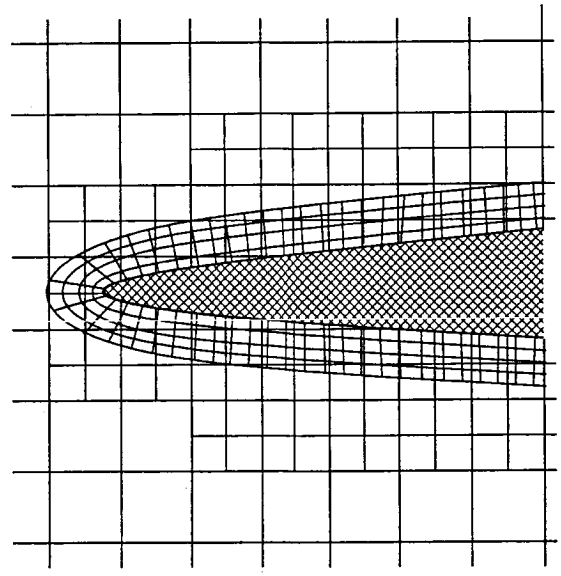


Figure 6: Example Chimera Grid Near Curved Surface

The development of chimera gridding schemes were not solely founded in the viscous/inviscid coupling problems, but chimera gridding schemes were applicable to that use. Throughout the history of chimera gridding there have been a number of motivations for their investigation such as increasing grid point resolution near solid bodies [13], overcoming structured gridding issues associated with modeling complex geometries for the full potential equation [14, 15, 55, 153] as well as the Euler equations [21, 109, 56], solving moving body problems [90, 100, 101] and resolving the boundary layers in Navier-Stokes calculations [78, 79, 180, 181, 182].

Atta [13] developed one of the first uses of chimera grids for the full potential equation

in two-dimensions using a finite difference formulation. A uniform Cartesian grid was used for the background grid and a body-fitted O-type structured grid was used around the body. The two grids were coupled via boundary information exchanges during the iteration process. First, the solution around the body fitted grid was converged through an outer iteration using a Dirichlet boundary condition imposed on the outer boundary. Next, the outer grid was converged using a Neumann boundary condition on the inner boundary, utilizing the solution information from the body solution. This information was then used to converge the body fitted grid once again. This cycle continued until the solution approached steady-state. This procedure required each grid (body and background) to have at least one complete cell inside the domain of the other, with the inner grid having an extent of between 1 and 3 chord lengths in all directions. Significant effort was needed to minimize the overlapping region in order to achieve optimal performance. Atta later extended this methodology to three-dimensions [14] as well as more complex configurations [15].

Steger et al. [153] developed a finite-difference chimera grid scheme that could handle a much larger variety of configurations compared to Atta's work. While limited to two-dimensions, they presented results for an airfoil-flap, cascading blades, a non-lifting bi-plane and an inlet with center body configuration. All of these configurations were handled automatically by their solver with little changes to the standard finite-difference formulations. State variables were exchanged between grids through interpolations which can cause performance penalties in the initialization stages when the connectivity is being constructed, but they addressed this by using the "stencil-walk" search pattern, where the

cells that are used for the interpolation of one cell are assumed to be close to the cells that are needed for the interpolation of that cell's neighbors.

A direct extension to the work of Steger et al. was developed by Benek et al. [21], named OVERFLOW, which applied chimera grid techniques to three dimensions and arbitrary body configurations as well as complete aircraft configurations. Meakin [102, 103] developed extensions that applied existing AMR techniques to the background meshes in order to resolve the off-body aerodynamics effects for Euler and Navier-Stokes equations. Additionally, Meakin developed techniques to apply AMR to unsteady, viscous, three-dimensional flows. In handling the viscous terms efficiently, the body-oriented grids were sized to capture the boundary layers, while the Cartesian grids were used for most of the computational domain. This resulted in an operation count drop of 2.5-6.5 with respect to the general curvilinear formulations (depending on whether the Euler, thin-layer Navier-Stokes or full Navier-Stokes equations were used). To further improve the handling of the viscous terms, the thin-layer Navier-Stokes equations could be used on the body-oriented grids since they were aligned with the dominant viscous stresses. This work provided the potential for significant floating point operation count reductions which resulted in an efficient solution technique. An excellent description of the modeling of a complex configuration was performed by Pearce et al. [125] where OVERFLOW was used to model the complete Space Shuttle Launch Vehicle.

Other interesting applications of chimera gridding was the use of all Cartesian meshes in the chimera grids by Mitcheltree et al. [109], and the use of multigridding techniques by

Epstein et al. [55, 56] as well as Kao et al. [78].

Hybrid Grid Schemes

Another approach that was related to the chimera grid approach was the use of unstructured grids between the body surface and the background Cartesian mesh, as opposed to the overlaying of these grids. These schemes were usually referred to as hybrid grid techniques. Figure 7 demonstrates an example hybrid grid around a curved surface in two dimensions.

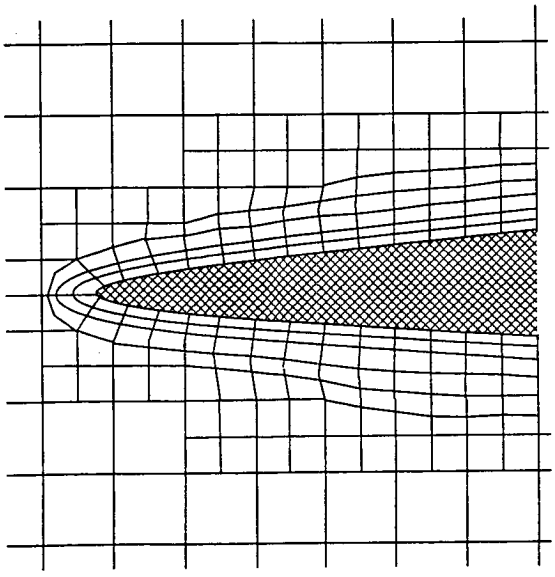


Figure 7: Example Hybrid Grid Near Curved Surface

One application of a hybrid scheme known as SPLITFLOW, by Karman [79] and enhanced by Domel and Karmen [52], used Cartesian grids for the majority of the computational domain, and prismatic grids to resolve the boundary layers. Standard Cartesian grid cutting techniques were used at the interface between the prismatic grids and the Cartesian

grid. The prismatic cells were grown from the surface triangulation using a marching layers technique [77]. Delanaye et al. [49] addressed significant difficulties that could arise in the prismatic-Cartesian technique near convex regions, overlapping regions, and other regions where the prismatic marching technique needed to be modified to create viable grids. Another similar effort to SPLITFLOW was performed by Wang [180, 182] except that instead of body oriented triangles or prismatic cells, body oriented quadrilateral cells were used to better capture the anisotropic nature of the viscous boundary layer regions.

Other Related Method

Similar to the reconstruction method is the class of finite element solution techniques called element-free Galerkin methods. Originally developed by Belytschko et al. [20] for elasticity and heat conduction problems, it is currently being investigated for its applicability to fluid dynamics [192] because of its automated handling of grid generation. The basic premise of this method is the use of polynomial curve fits to approximately represent the data surrounding the node of interest. Typically, a least-squares error minimization is used due to the larger number of data points surrounding the node than the number of unknowns in the curve fit. Most implementations demonstrate oscillations near sharp gradients (especially with higher-order interpolation functions) with more research needed to developing effective limiters.

Another scheme related to the reconstruction method that is the gridless method originally developed by Batina [18]. This method uses a cloud of points to reconstruct a polynomial curve fit (similar to the element-free Galerkin method) using a least-squares error

minimization. These curve fits are then used to calculate the derivatives required to solve the Navier-Stokes equations in differential form. The number of calculations per node is higher than for other techniques due to the large number of least-squares fits that are required. Unfortunately, this scheme does is not conservative and requires numerical dissipation in order to obtain a solution. Other researchers have extended this work [91], but without addressing the conservation problem.

Parallelization Efficiency Approaches

Parallelization efforts throughout the history of CFD have been strongly influenced by the computational hardware available to the researchers. In the early years of CFD, the dominant hardware available to researchers was SIMD (Single Instruction Multiple Data) architectures. These were also known as vector based architectures, and they used long vectors of data (with the size depending on the size of the computer's pipeline) and performed the same operation on each data item in the pipeline in a single CPU clock cycle. Different operations could be chained together to create an assembly line of operations without having to use excess cycles to fill the pipeline caches on each arithmetic unit. Thus, it took the same amount of time to perform 64 multiplies as it would 1 multiply on a vector machine with a pipeline size of 64 or larger. While the SIMD architectures provided excellent parallelization potential on problems with long vectors of data, they became of limited use to current large CFD applications because of the expensive memory that was required for these architectures as well as the rise of other less costly architectures [96].

The main parallelization architectures that took the place of the SIMD architectures was the MIMD (Multiple Instruction Multiple Data) architectures. These architectures utilized multiple processors to process the data in parallel using possibly different sets of computer instructions on each piece. Thus, it was possible to perform two independent tasks concurrently and not be restricted to the vector paradigm in the algorithm development as in the SIMD architectures.

SIMD Parallelization

Most early CFD work on SIMD architectures, such as [23, 105, 152] focused on achieving results quickly without quantitative analysis of the parallelization performance. Discussions typically provided wall clock results for the cases demonstrated, but no comparison was usually offered between scalar and vector runs nor was there any comparison between various sized pipelines. Heller [66] provided a table of selected timings for common operations on the CDC STAR SIMD architecture that provided useful timing information for predicting performance characteristics for a given set of operations on a data vector. References [132] and [175] provide additional information about vectorization and how to prepare code for vectorization.

MIMD Parallelization

MIMD architectures are generally split into two classes depending on the connectivity used between processors. The first is the shared memory based architectures where all of the memory is available to each processor in one common address space. shared memory

architectures usually consist of a number of CPUs connected to a common block of memory that was addressable to all processors. Each processor may also have its own separate memory (such as on die caches or memory modules separate from the common banks), but that memory was not part of the shared memory collective. Most current shared memory architectures provide a hierarchy of physical memory locations the have varying access timings such that there is a certain amount of locality associated with memory accesses. These architectures, known as cache-coherent Non-Uniform Memory Architectures or cc-NUMA, require the application to address this memory locality issue in order to obtain maximum performance. Parallelization in these environments can efficiently be performed using common programming techniques such as shared memory structures and light-weight threads to perform the parallel tasks on separate processors with little overhead involved in exchanging information between the parallel tasks.

The other MIMD architecture is the distributed memory based architecture where each processor has its own local memory address space that is not shared with the other processors. Distributed memory architectures consist of a collection of CPUs that each contain their own memory modules with no direct connectivity to the other CPUs memory, and thus the memory of another processor is not directly addressable across the processor boundary. This architecture does not allow for simple, efficient implementations of the same parallel programming techniques typical of shared memory architectures. Specifically, there is no simple way of handling shared memory structures, nor is there a way of efficiently spawning threads on separate processors and keeping all of the shared data synchronized between

each processor's memory. Thus, information to be shared between parallel tasks needs to be explicitly exchanged between the tasks in a much more controlled and orderly fashion. Frequently, this is handled by using standard client-server communication paradigms such as message passing.

Heller [66] and Voigt [175] provided an excellent discussion of general parallelization schemes that could be utilized in MIMD architectures, while Venkatakrishnan [172] provided an informative section on the parallelization issues associated with MIMD architectures and CFD. Wang [179] provided a comparison of the parallelization performances of several systems including Cray T3D and T3E [43] shared memory architectures and a Beowulf [157, 80] distributed memory system with results that indicated comparable speedups for all architectures as long as the amount of communication was much less than the amount of computation.

Parallelization Libraries

In recent years, three major standard libraries have been used extensively in the parallelization of CFD applications on MIMD architectures, OpenMP [121, 122], MPI [106, 107] and PVM [61]. While all three libraries provide unique benefits, only a comparison between OpenMP and MPI will be presented.

OpenMP is a parallelization library that was specifically designed for shared memory architectures. It allows for incremental parallelization of existing applications and utilizes many shared memory features to optimize its performance (such as shared memory information exchange, light-weight threads, and operating system level signals and

semaphores). It provides coarse grain as well as fine grain parallelization mechanisms, and it is compatible with FORTRAN, C, and C++ programming languages on a variety of hardware and operating system combinations. However, it currently can not efficiently utilize distributed memory parallel hardware because of its intricate dependency on the shared memory paradigm. Thus there is an entire class of parallel hardware that the OpenMP based applications can not support easily.

MPI is a parallelization application programming interface (API) that is based on the idea of parallel tasks communicating using either synchronous or asynchronous message exchanges. MPI can be used in both shared and distributed memory architectures, and supports FORTRAN, C, and C++ programming languages on a wide range of hardware and operating system combinations. Additionally, MPI does not exclude the use of a heterogeneous collection of hardware and operating systems, thus it allows for an extremely diverse configuration to be utilized in a distributed memory parallel fashion. However, the MPI API does not specifically handle such issues as byte-ordering, data representation differences, or data sizes, this has to be handled by the application. In a shared memory environment, the message passing paradigm creates an added overhead to the parallel task communication process due to the need to pack, send, receive, and unpack all information exchanges. Most MPI implementations optimize the communication on shared memory nodes by replacing the send-receive portion of the message passing operation with the use of a common shared memory cache. Additionally, MPI does not provide the same level of incremental parallelization that OpenMP provided. Jespersen [76] provided an overview

of the message passing schemes needed for OVERFLOW (a large scale CFD application) using MPI.

Shared Memory Based Schemes

There are currently two main CPU-memory interconnection schemes that are used in shared memory architectures, bus-based and switch-based. The bus-based architecture have a relatively narrow bandwidth connection between the CPUs that could easily become saturated if too many memory access requests occur. Thus, this architecture is limited in its scalability. The other type of interconnection is the switched-based architecture. This architecture provides more of a matrixed connectivity between the CPUs and the memory modules, as well as provides multiple paths for memory accesses to travel and reduces the bandwidth limitations seen in the bus-based approach. Reference [123] provides an excellent review of these topics. With the increased connectivity speeds of networking technologies, research into providing a shared memory interface on top of a distributed memory architecture has been performed, see reference [139] for more details.

The high performance improvements that Aftosmis et al. [2, 3] developed for their shared memory based CART3D solver, see page 7 for more information, mainly focused on the preprocessing steps that were performed before the actual solution code was run. In order to improve the parallelization speedup of their code, they developed a set of cell reordering techniques that used a concept called space-filling curves [144] to minimize the inter-process communication due to the domain decomposition. The space-filling curves also provided an optimal ordering of the data on each node that maximized the on-board

cache usage on each processor and were utilized in every stage of the multigrid solution cycle, which created slightly more communication overhead, but ensured load balancing on all multigrid stages. The other major improvement made was a transformation of the adaptive refinement techniques from floating point mathematics to integer based mathematics. This allowed them to utilize geometry calculation techniques from the field of computer graphics [37, 176] to perform the surface intersection tests using only a few machine clock-cycles per test. The overall parallelization of their code was done using OpenMP, and its performance achieved a nearly linear speedup for up to 64 processors, with parallel efficiencies (a measure of how efficiently the solver performed for n processors, defined as $E_n = \frac{T_{1proc}}{nT_{nprocs}}$) of approximately 0.9. An excellent summary of these performance improvements was in references [4] and [22].

Another shared memory based CFD solver was an unstructured, three-dimensional turbulent Navier-Stokes solver developed by Mavriplis [99, 96] that used a Runge-Kutta explicit time solver in a multigrid algorithm. In addition, directional smoothing and coarsening techniques were used to address the stiffness associated with high aspect-ratio cells. The computational domain was partitioned is such a way as to minimize the inter-grid data dependencies in the tri-diagonal solver associated with the directional smoothing. Impressive parallelization speedups were achieved for a variety of parallel architectures using the single grid scheme, including parallel efficiencies of 0.9 for a Cray T3E using 1450 nodes and the ASCI Red machine, with lower efficiencies for V- and W-Cycle multigrid cases due to the added communication overhead associated with the lower points per node

distribution of the coarser grid.

Sharov et al. [148] developed a shared memory based CFD solver that optimized the performance on cached-based parallel computers by using a variety of grid partitioning schemes. In addition to the space-filling curve reordering mentioned above, they also utilized a wavefront renumbering [92]. They also paid special attention to the parallelization of the GMRES preconditioner in order to optimize performance. Their results indicated that the space-filling curves provided the best grid reordering with a parallel efficiency of 0.5 for 20 nodes on an SGI Origin 2000.

Distributed Memory Based Schemes

The interconnection mechanisms for distributed memory architectures typically are done by some type of high bandwidth networking, such as 10 Mb, 100 Mb, or gigabit ethernet. In addition to the connectivity bandwidth, there are several interconnection topologies that can be employed. There are fully connected networks where every node could directly communicate with every other node (which becomes difficult to maintain with large numbers of nodes), as well as hypercubes and meshes where the nodes are conceptually distributed in multiple dimensions and then connected to their nearest neighbors (which limits the connectivity for each node, but can require a large number of hops to traverse the entire network), and also there are rings and linear arrays where the connectivity to each node is limited to 1 or 2 neighbors and traversing the network required sequential hops along the nodes (which is a simple network topology, but created only 1 or 2 paths for communications to travel and easily leads to network saturation). References [123] and [68]

provide more information on these topologies and advances in the distributed memory architectures. One final evolving technology is the idea of creating low-latency connectivity by providing a near-fully connected network via multiple network interface cards at each node [50]. This technique provides extremely high communication bandwidth, but required a complicated wiring and network switching scheme.

Early distributed memory results were from Decker et al. [47]. They provided an excellent discussion of various parallelization schemes and their efficacy in implicit finite difference schemes. They investigated several data distribution schemes for their parallelization efforts and provided a timing estimation for each scheme. They also demonstrated parallelization efficiencies of 0.9 for block tridiagonal cases and 0.8 for penta-diagonal cases (both using 4, 9, and 16 processors).

Barth and Linton [17] provided another early distributed memory based parallelization effort for an implicit, unstructured, turbulent Navier-Stokes solver in three-dimensions. The computational domain used a variety of methods to perform an *a priori* partitioning of the grid into subdomains that reside on each processor [173]. Their results showed that the spectral partitioning method provided the best load-balancing, but it required the most computational time. They used MPI as their parallelization scheme and provided results for the IBM SP2 [151]. Barth and Linton reported acceptable scalability results up to 64 processors with parallel efficiencies around 0.8 and the total number of iterations required for convergence slightly increasing as the number of processors increased.

More recent work was performed by Wang that utilized GMRES/multigrid schemes [181]

to improve convergence, accuracy and distributed memory parallelization speedup on an IBM SP2 using MPI. Wang used two different domain decomposition techniques, Recursive Coordinate Bisection and Recursive Spectral Bisection [130, 150], and concluded that the Recursive Coordinate Bisection method was superior due to its ability to create better load-balanced domains quickly at the expense of producing slightly more interface cells between domains. Additionally, domain decomposition occurred on the coarsest grid, so all finer grids in the multigrid cycle were required to exist on the same processor as the parent in order to eliminate the additional communication overhead mentioned above with Aftosmis et al. on page 35. Wang's results showed good parallelization performance for up to 16 processors (with the parallel efficiencies of 0.7), at which point each processor had few computational cells, and the communication costs overwhelmed the parallelization improvements. Wang also provided a scheme for improving parallelization efficiency by using a Communication and Computation Overlap procedure that reordered the computational cells such that the interior cells were being computed while the boundary cells were being exchanged between processors. This resulted in a savings of 10% to 20%.

Wu and Zou [187] provided a distributed memory based parallelization scheme for the two-dimensional steady and unsteady Euler equations using PVM as outlined in reference [143]. Their work focused on the use of overlapping grids in order to independently solve the equations on each grid. This required time-lagging of the overlapping grids, and a discussion was presented for the use of various time-lagging schemes. The resulting schemes produced reasonable parallelization efficiencies for most time-lagging schemes,

with the most consistent results occurring when the entire overlapping grid data was at the previous time step, as opposed to it being two time steps back or only lagging the implicit portions of their scheme.

Venkatakrishnan [170, 171] provided an excellent discussion on distributed memory parallelization issues for solving the two-dimensional flow problems using explicit and implicit formulations. Eidson and Erlebacher [54] presented a detailed description of the implementation issues that resulted from solving a periodic tridiagonal linear system (a common linear system in CFD) which provided significant implementation details that can be of use for other linear system solvers.

Combined Approaches

One final MIMD parallelization effort worth noting was a combination of the shared and distributed memory based schemes. Mavriplis [97, 98] developed a combination OpenMP and MPI unstructured grid solver [96, 99] based on his research discussed above on page 35. This scheme utilized MPI communication techniques for distributed memory parallelization tasks and OpenMP communication techniques for shared memory parallelization tasks. This was an effort to optimize performance on shared memory architectures that resided in a distributed memory network. For the architectures that he evaluated, the MPI alone and OpenMP alone versions produced similar parallelization results on shared memory architectures, and the MPI alone version performed better than the hybrid OpenMP and MPI version for a cluster of shared memory machines.

Scope of Current Work

As has been mentioned above, the current approaches to modeling the Navier-Stokes equations on Cartesian grids have difficulties near the cut cells. Additionally, the requirements put on the numbers of grid cells needed near the solid surfaces in order to accurately resolve the viscous effects make the use of traditional solid surface boundary condition treatments inadequate to efficiently solve the Navier-Stokes equations on full aerodynamic configurations. The grid cell resolution issues also make the use of Cartesian grid schemes on a single computer unrealistic for full aerodynamic configurations due to the large numbers of computational cells (10's of millions) and the long computational times (hours or even days) required to achieve a practical solution. Thus a strategy must be developed that addresses these major difficulties in Navier-Stokes Cartesian solvers if they are ever to gain widespread use.

This thesis presents two extensions to Cartesian grid solution functionalities. First is a scheme for modeling the compressible three-dimensional Navier-Stokes equations in a Cartesian solver by using an interpolation based boundary condition for the surface cells in order to avoid the non-smoothness associated with the schemes investigated by Coirier [38] mentioned above. This technique has the added benefit of removing the cut cells from the time step restriction associated with traditional schemes. The second enhancement is a distributed memory parallelization port of an existing Cartesian solver in order to utilize the solver on a larger variety of parallel processing environments.

Traditional boundary condition approaches use Taylor series based approximations of one-sided differencing and no-slip boundary conditions for viscous and heat flux calculations. The current research utilizes an interpolation based scheme which utilizes the existing boundary conditions along with the governing equations to update the state vector for the computational cells that are on the body surface. This removes the surface cells from the finite volume formulation, and thus removes the time step restriction associated with the arbitrarily small cut cells. It also provides an alternative to the cell merging techniques that other Cartesian schemes use to address the cut cell time step restriction. This scheme is implemented within an existing three-dimension finite volume Cartesian grid solver where the traditional second order numerical differences are applied to the off-body terms, and this new scheme is applied in the solid wall boundary cells.

To address the increased computational costs associated with Navier-Stokes Cartesian grid solvers, Cartesian solvers need to be able to utilize the growing numbers of inexpensive commercial off-the-shelf distributed memory parallel computing environments. Using standard networking components and techniques, a high-speed distributed memory computational environment can be created that competes with more expensive shared memory architectures on certain tasks for a fraction of the costs. If implemented properly, Computational Fluid Dynamics can be one of those tasks, since interprocess communication (IPC) in parallel CFD is a relatively low bandwidth task. The major effort associated with utilizing this new parallel environment for existing CFD applications is to take the existing shared memory parallel codes and convert them to distributed memory parallel codes. By

isolating the IPC tasks from the CFD tasks in the code, identifying similar parallel tasks in each paradigm and eliminating the usage of techniques that are exclusive to shared or distributed memory parallelization, an efficient solver has been created that can be utilized in a shared or distributed memory environment with little impact on the overall parallelization performance.

In the present thesis, Chapter II provides a description of the Cartesian solvers that are being investigated throughout this research. A detailed description of the newly created Cartesian solver NASCART-GT is presented. This is followed by an overview of the existing solver, CART3D, with descriptions of the important functionalities and capabilities.

Chapter III provides a description of the new solid boundary treatment for both inviscid and viscous flows. It starts with a description of the limitations of the current procedures for viscous flux reconstructions at the solid boundary. It then puts forward an alternative treatment of the solid boundary cells that avoids the deficiencies associated with the current solid boundary cell treatments.

The next chapter, Chapter IV, describes the development of the parallelization enhancements made to CART3D. Specific details are given that describe the changes that were made to the existing code as well as the code additions that were made.

Chapters V and VI provides the results due to these improvements. First, simple geometry results are presented for inviscid cylinder and viscous flat plate flows in order to examine the improvements for cases that have well known analytical solutions. Next, simple aerodynamic geometries are presented for transonic inviscid as well as subsonic and

supersonic viscous flows around a NACA-0012 airfoil. These cases demonstrate the effectiveness of these schemes for aerodynamic configurations which have well studied experimental and numerical solutions. This is followed by an demonstration of the effectiveness of the improvements for a transonic inviscid flow over an ONERA-M6 wing. Finally, results are presented demonstrating the parallelization improvements compared to existing shared memory results, as well as parallelization results for a distributed memory architecture.

Finally, Chapter VII presents a summary of the conclusions obtained from this research as well as some suggestions for future development.

CHAPTER II

EXISTING CARTESIAN GRID SOLVERS

A summary is presented of the current functionality of the Cartesian solvers that were modified in order to provide an understanding of the starting point for this research. CART3D is a well established Cartesian solver used for a number of problems, see [4] and [124], and provides capabilities of solving 3D, compressible, inviscid flows. CART-GT was developed recently and provides capabilities of solving 3D, compressible inviscid flows as well as viscous flows with traditional finite differencing of the viscous terms.

NASCART-GT

NASCART-GT is an unsteady, three-dimensional Cartesian grid solver of the full Navier-Stokes equations without body forces and a perfect gas thermodynamic model. The Navier-Stokes equations are solved using Roe's approximate Riemann solver coupled with a MUSCL data reconstruction technique for the inviscid fluxes and traditional finite differencing of the viscous terms. In all this creates a second order spatially accurate scheme. The time integration is performed using a Hancock two-stage predictor-corrector scheme which is second order accurate in time. In order to accurately capture high gradient regions, a solution adaption scheme is used that is uses the velocity divergence as the coarsening/refining

metric.

Governing Equations

The three-dimensional Navier-Stokes equations are the governing equations solved in NASCART-GT, shown in the integral form in equations (1a)–(1c).

$$\frac{\partial}{\partial t}\iiint_{CV} \rho \, d\mathcal{V} + \iint_{CS} \rho \, (\boldsymbol{v}\cdot\boldsymbol{n}) \, dA = 0 \tag{1a}$$

$$\frac{\partial}{\partial t}\iiint_{CV} \rho \boldsymbol{v} \, d\mathcal{V} + \iint_{CS} \rho \boldsymbol{v} \, (\boldsymbol{v}\cdot\boldsymbol{n}) \, dA = -\iint_{CS} p \, \boldsymbol{n} \, dA + \iint_{CS} [\tau]\boldsymbol{n} \, dA + \iiint_{CV} \rho \frac{d\boldsymbol{F}_b}{dm} \, d\mathcal{V}$$
$$- \iiint_{CV} \rho \left[\frac{d^2\boldsymbol{r}}{dt^2} + \frac{d\Omega}{dt}\cdot\boldsymbol{r} + \Omega\times(\Omega\times\boldsymbol{r}) + 2\Omega\times\boldsymbol{v} \right] d\mathcal{V} \tag{1b}$$

$$\frac{\partial}{\partial t}\iiint_{CV} \rho e_t \, d\mathcal{V} + \iint_{CS} \rho e_t \, (\boldsymbol{v}\cdot\boldsymbol{n}) \, dA = -\iint_{CS} p \, (\boldsymbol{v}\cdot\boldsymbol{n}) \, dA$$
$$+ \iint_{CS} \boldsymbol{v}^T [\tau]\cdot\boldsymbol{n} \, dA + \iint_{CS} k\nabla T\cdot\boldsymbol{n} \, dA \tag{1c}$$

where $\quad e_t = e_{internal} + \dfrac{\boldsymbol{v}\cdot\boldsymbol{v}}{2} + \boldsymbol{h}\cdot\boldsymbol{g}_{\oplus}$

and $\quad [\tau] = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix}$

With the components of the viscous stress tensor given by

$$\begin{aligned}
\tau_{xx} &= \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) \\
\tau_{yy} &= \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \\
\tau_{zz} &= \frac{2}{3}\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \\
\tau_{xy} &= \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\
\tau_{xz} - \tau_{zx} &= \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\
\tau_{yz} &= \tau_{zy} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)
\end{aligned} \tag{2}$$

and $\boldsymbol{h}$ being a height above an arbitrary datum. By allowing no body forces, assuming that the elevation changes within the flow field are negligible and assuming the control volume is stationary, equations (1a)–(1c) become

$$\frac{\partial}{\partial t}\iiint_{CV} \rho \, d\mathcal{V} + \iint_{CS} \rho\,(\boldsymbol{v}\cdot\boldsymbol{n})\, dA = 0 \tag{3a}$$

$$\frac{\partial}{\partial t}\iiint_{CV} \rho \boldsymbol{v} \, d\mathcal{V} + \iint_{CS} \rho \boldsymbol{v}\,(\boldsymbol{v}\cdot\boldsymbol{n})\, dA = -\iint_{CS} p\,\boldsymbol{n}\, dA + \iint_{CS} [\tau]\boldsymbol{n}\, dA \tag{3b}$$

$$\begin{aligned}
\frac{\partial}{\partial t}\iiint_{CV} \rho e_t \, d\mathcal{V} + \iint_{CS} \rho e_t\,(\boldsymbol{v}\cdot\boldsymbol{n})\, dA &= -\iint_{CS} p\,(\boldsymbol{v}\cdot\boldsymbol{n})\, dA \\
&\quad + \iint_{CS} \boldsymbol{v}^T [\tau]\cdot\boldsymbol{n}\, dA + \iint_{CS} k\nabla T\cdot\boldsymbol{n}\, dA
\end{aligned} \tag{3c}$$

where $\quad e_t = e_{internal} + \dfrac{\boldsymbol{v}\cdot\boldsymbol{v}}{2}$

with equation set (2) still holding for the stress tensor elements. By defining the state vector as

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix} \tag{4}$$

equations (3a)–(3c) can be rewritten as

$$\frac{\partial}{\partial t}\iiint_{CV} \boldsymbol{U}\, d\mathcal{V} + \iint_{CS} (\mathcal{F}_I - \mathcal{F}_V)\cdot \boldsymbol{n}\, dA = 0 \tag{5}$$

with the inviscid and viscous fluxes defined as

$$\mathcal{F}_I = \begin{bmatrix} \rho \boldsymbol{v} \\ \rho u \boldsymbol{v} + p\boldsymbol{i} \\ \rho v \boldsymbol{v} + p\boldsymbol{i} \\ \rho w \boldsymbol{v} + p\boldsymbol{k} \\ \rho (e_t + p)\boldsymbol{v} \end{bmatrix} \qquad \mathcal{F}_V = \begin{bmatrix} 0 \\ \tau_x \\ \tau_y \\ \tau_z \\ (u\tau_x + v\tau_y + w\tau_z) + k\nabla T \end{bmatrix} \tag{6}$$

In order to close the system of equations, a thermodynamic model needs to be used. The thermodynamic model used in NASCART-GT is a calorically perfect gas model with the standard equation of state given by equation (7).

$$p = \rho RT \tag{7}$$

For calorically perfect gas, the following relationships hold

$$e = C_v T \quad h = C_p T \quad \gamma = \frac{C_p}{C_v} \quad C_v = \frac{R}{\gamma - 1} \quad C_p = \frac{\gamma R}{\gamma - 1} \tag{8}$$

Additionally, models need to be established for the transport properties. By assuming a constant Prandtl number and Sutherland's formula for the viscosity model, the following equations are used for the dynamic viscosity and thermal conductivity, respectively

$$\mu = C_1 \frac{T^{3/2}}{T + C_2} \tag{9a}$$

$$k = \frac{C_p \mu}{Pr} \tag{9b}$$

where C_1 and C_2 are constants for a given gas.

To actually perform the calculations, the equations (4), (5), and (6) are non-dimensionalized using a characteristic length, l, and the freestream density, ρ_∞, velocity, V_∞, and dynamic viscosity, μ_∞. These can be combined to form the Reynolds number $Re_\ell = \frac{\rho_\infty V_\infty l}{\mu_\infty}$. The following equations are the result of the non-dimensionalization

$$\boldsymbol{U} = \begin{bmatrix} \rho^* \\ \rho^* u^* \\ \rho^* v^* \\ \rho^* w^* \\ \rho^* e_t^* \end{bmatrix} \tag{10}$$

$$\text{where} \qquad e_t^* = e_{internal}^* + \frac{\boldsymbol{v}^* \cdot \boldsymbol{v}^*}{2}$$

$$\frac{\partial}{\partial t^*} \iiint_{CV} \boldsymbol{U}^* \, d\mathcal{V}^* + \iint_{CS} (\mathcal{F}_I^* - \mathcal{F}_V^*) \cdot \boldsymbol{n}^* \, dA^* = 0 \tag{11}$$

$$\mathscr{F}_I^* = \begin{bmatrix} \rho v^* \\ \rho u v^* + p^* \boldsymbol{i} \\ \rho v v^* + p^* \boldsymbol{j} \\ \rho w v^* + p^* \boldsymbol{k} \\ \rho \left(e_t^* + p^*\right) v^* \end{bmatrix} \qquad \mathscr{F}_V^* = \begin{bmatrix} 0 \\ \tau_x^* \\ \tau_y^* \\ \tau_z^* \\ \left(u^* \tau_x^* + v^* \tau_y^* + w^* \tau_z^*\right) + \frac{k^* \nabla T^*}{Pr\, Re_\ell\, M_\infty^2} \end{bmatrix} \tag{12}$$

with the following non-dimensionalizations

$$x^* = \frac{x}{\ell} \quad y^* = \frac{y}{\ell} \quad z^* = \frac{z}{\ell} \quad t^* = \frac{t}{\ell / V_\infty} \quad u^* = \frac{u}{V_\infty} \quad v^* = \frac{v}{V_\infty} \quad w^* = \frac{w}{V_\infty} \tag{13}$$

$$\rho^* = \frac{\rho}{\rho_\infty} \quad p^* = \frac{p}{\rho_\infty V_\infty^2} \quad T^* = \frac{T}{p_\infty / (\rho_\infty R)} \quad e^* = \frac{e}{V_\infty^2} \quad \mu^* = \frac{\mu}{\mu_\infty} \quad k^* = \frac{k}{\gamma R \mu_\infty / Pr}$$

with the viscous terms non-dimensionalized as

$$\begin{aligned}
\tau_{xx}^* &= \frac{\tau_{xx}}{\rho_\infty V_\infty^2} = \frac{2}{3} \frac{\mu^*}{Re_\ell} \left(2 \frac{\partial u^*}{\partial x^*} - \frac{\partial v^*}{\partial y^*} - \frac{\partial w^*}{\partial z^*} \right) \\
\tau_{yy}^* &= \frac{\tau_{yy}}{\rho_\infty V_\infty^2} = \frac{2}{3} \frac{\mu^*}{Re_\ell} \left(2 \frac{\partial v^*}{\partial y^*} - \frac{\partial u^*}{\partial x^*} - \frac{\partial w^*}{\partial z^*} \right) \\
\tau_{zz}^* &= \frac{\tau_{zz}}{\rho_\infty V_\infty^2} = \frac{2}{3} \frac{\mu^*}{Re_\ell} \left(2 \frac{\partial w^*}{\partial z^*} - \frac{\partial u^*}{\partial x^*} - \frac{\partial v^*}{\partial y^*} \right) \\
\tau_{xy}^* &= \frac{\tau_{xy}}{\rho_\infty V_\infty^2} = \frac{\mu^*}{Re_\ell} \left(\frac{\partial u^*}{\partial y^*} + \frac{\partial v^*}{\partial x^*} \right) \\
\tau_{xz}^* &= \frac{\tau_{xz}}{\rho_\infty V_\infty^2} = \frac{\mu^*}{Re_\ell} \left(\frac{\partial u^*}{\partial z^*} + \frac{\partial w^*}{\partial x^*} \right) \\
\tau_{yz}^* &= \frac{\tau_{yz}}{\rho_\infty V_\infty^2} = \frac{\mu^*}{Re_\ell} \left(\frac{\partial v^*}{\partial z^*} + \frac{\partial w^*}{\partial y^*} \right) \\
k^* \nabla T^* &= \frac{Pr\, Re_\ell\, M_\infty^2}{\rho_\infty V_\infty^3} k \nabla T
\end{aligned} \tag{14}$$

Inviscid Flux Calculations

The inviscid fluxes in NASCART-GT are calculated using the well known Roe's approximate Riemann solver coupled with a MUSCL data reconstruction technique. More information about Roe's approximate Riemann solver can be found in references [141, 142, 159], and more information about the MUSCL data reconstruction technique can be found in references [158, 159].

Roe's Approximate Riemann Solver

In order to accurately capture the physical effects modeled by the fluid dynamics equations, it is important to discretize the equations in the direction of information propagation. One method of capturing this phenomena is the Flux Difference Splitting technique which models the flow phenomena as a collection of local wave propagation between control volumes, also known as the Godunov approach[62, 63]. Roe's approximate Riemann solver belongs to this class of solution procedures, details of which can be found in references [141, 142] with implementation details in [159, 177].

Roe's method provides a method of calculating the flux across a face of a control volume using the eigenvalues, λ_i, the right eigenvectors, $\boldsymbol{K}_i$, and the wave strengths, α_i. Equations (15), (17), and (16) show how the flux is calculated for the an x-face.

$$\boldsymbol{F}_{l+\frac{1}{2}} = \frac{1}{2}\left[(\boldsymbol{F}_L + \boldsymbol{F}_R) - \sum_{i=1}^{5} \tilde{\alpha}_i |\tilde{\lambda}_i| \tilde{\boldsymbol{K}}_i\right] \tag{15}$$

where $\boldsymbol{F}_L$ is the flux calculated using the left state vector and $\boldsymbol{F}_R$ is the flux calculated using

the right state vector and

$$\tilde{\alpha}_1 = \frac{\Delta p - \tilde{\rho}\tilde{a}\Delta u}{2\tilde{a}^2} \quad \tilde{\alpha}_2 = \Delta\rho - \frac{\Delta p}{\tilde{a}^2} \quad \tilde{\alpha}_3 = \tilde{\rho}\Delta v \quad \tilde{\alpha}_4 = \tilde{\rho}\Delta w \quad \tilde{\alpha}_5 = \frac{\Delta p + \tilde{\rho}\tilde{a}\Delta u}{2\tilde{a}^2}$$

$$\tilde{\lambda}_1 = \tilde{u} - \tilde{a} \quad \tilde{\lambda}_2 = \tilde{u} \quad \tilde{\lambda}_3 = \tilde{u} \quad \tilde{\lambda}_4 = \tilde{u} \quad \tilde{\lambda}_5 = \tilde{u} + \tilde{a} \tag{16}$$

$$\tilde{K}_1 = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} - \tilde{u}\tilde{a} \end{bmatrix} \quad \tilde{K}_2 = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \frac{\tilde{V}^2}{2} \end{bmatrix} \quad \tilde{K}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \tilde{v} \end{bmatrix} \quad \tilde{K}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \tilde{w} \end{bmatrix} \quad \tilde{K}_5 = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} + \tilde{u}\tilde{a} \end{bmatrix}$$

with $\tilde{H} = \frac{\tilde{V}^2}{2} + \tilde{e} + \frac{\tilde{p}}{\tilde{\rho}}$ and $\tilde{V}^2 = \tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2$

with the average state calculated as

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \\
\tilde{u} &= \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{v} &= \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{w} &= \frac{\sqrt{\rho_L}w_L + \sqrt{\rho_R}w_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{a} &= \sqrt{(\gamma - 1)\left(\tilde{H} - \frac{\tilde{V}^2}{2}\right)}
\end{aligned} \tag{17}$$

Applying equations (15), (17) and (16) to a Cartesian control volume results in the following formulation for the flux across a face

$$\mathscr{F} = \frac{1}{2}(\mathscr{F}_L + \mathscr{F}_R + \Theta) \tag{18}$$

where
$$\Theta = -|\tilde{\phi}| \left\{ \left(\Delta\rho - \frac{\Delta p}{\tilde{a}^2}\right) \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \frac{\tilde{V}^2}{2} \end{bmatrix} + \tilde{\rho} \begin{bmatrix} 0 \\ \Delta u - n_x \Delta\phi \\ \Delta v - n_y \Delta\phi \\ \Delta w - n_z \Delta\phi \\ \Delta\left(\frac{V^2}{2}\right) - \tilde{\phi}\Delta\phi \end{bmatrix} \right\}$$

$$- |\tilde{\phi} + \tilde{a}| \frac{\Delta p + \tilde{\rho}\tilde{a}\Delta\phi}{2\tilde{a}^2} \begin{bmatrix} 1 \\ \tilde{u} + n_x\tilde{a} \\ \tilde{v} + n_y\tilde{a} \\ \tilde{w} + n_z\tilde{a} \\ \tilde{H} + \tilde{\phi}\tilde{a} \end{bmatrix}$$

$$- |\tilde{\phi} - \tilde{a}| \frac{\Delta p - \tilde{\rho}\tilde{a}\Delta\phi}{2\tilde{a}^2} \begin{bmatrix} 1 \\ \tilde{u} - n_x\tilde{a} \\ \tilde{v} - n_y\tilde{a} \\ \tilde{w} - n_z\tilde{a} \\ \tilde{H} - \tilde{\phi}\tilde{a} \end{bmatrix}$$

For the flux calculation in the x-direction: $\phi = u$, $\tilde{\phi} = \tilde{u}$, $n_x = 1$, $n_y = n_z = 0$, $\mathscr{F} = \boldsymbol{F}_x$ and L/R vary in the x-direction. For the flux calculation in the y-direction: $\phi = v$, $\tilde{\phi} = \tilde{v}$, $n_y = 1$, $n_x = n_z = 0$, $\mathscr{F} = \boldsymbol{F}_y$ and L/R vary in the y-direction. For the flux calculation in the

z-direction: $\phi = w$, $\tilde{\phi} = \tilde{w}$, $n_z = 1$, $n_x = n_y = 0$, $\mathscr{F} = \boldsymbol{F}_z$ and L/R vary in the z-direction.

MUSCL Data Reconstruction

The MUSCL (Monotone Upstream-centered Scheme for Conservation Laws) data reconstruction scheme originated with van Leer [164, 165, 166] introducing a piece-wise linear reconstruction of the primitive state variable instead of the piece-wise constant reconstruction used in lower order Godunov schemes. The reconstructed data can be plugged into a flux reconstruction scheme, such as equations (17) and (18) to produce the inviscid fluxes. Equations (19) and (20) shows a MUSCL reconstruction for the $i+\frac{1}{2}$ face of a Cartesian control volume.

$$\boldsymbol{W}_{L_{i+\frac{1}{2},j,k}} = \boldsymbol{W}_{i,j,k} + \frac{\varepsilon_{i,j,k}}{4}\left[(1-\kappa)\left(\boldsymbol{W}_{i,j,k} - \boldsymbol{W}_{i-1,j,k}\right) + (1+\kappa)\left(\boldsymbol{W}_{i+1,j,k} - \boldsymbol{W}_{i,j,k}\right)\right] \quad (19)$$

$$\boldsymbol{W}_{R_{i+\frac{1}{2},j,k}} = \boldsymbol{W}_{i+1,j,k} - \frac{\varepsilon_{i,j,k}}{4}\left[(1+\kappa)\left(\boldsymbol{W}_{i+1,j,k} - \boldsymbol{W}_{i,j,k}\right) + (1-\kappa)\left(\boldsymbol{W}_{i+2,j,k} - \boldsymbol{W}_{i+1,j,k}\right)\right] \quad (20)$$

where

$$\boldsymbol{W} = \begin{bmatrix} \rho \\ u \\ v \\ w \\ H \end{bmatrix} \quad (21)$$

and $\varepsilon_{i,j,k} = 0$ is traditional first order piece-wise constant and $\varepsilon_{i,j,k} = 1$ is second or third order (depending on the value of κ). For the $\varepsilon_{i,j,k} = 1$ cases, if $\kappa = -1$ use second order fully upwind biased scheme, if $\kappa = 1/3$, then use third order upwind biased scheme, if $\kappa = 0$ then use second order upwind biased scheme and if $\kappa = 1$ then use second order central difference scheme. Details about the population of the neighboring cells is discussed on page 58. Reconstructing the other 5 faces follows in a similar fashion.

The Monotonicity of the scheme is introduced via a limiter that sets the data reconstruction to first order in regions of high pressure gradients using the following

$$\varepsilon_{i,j,k} = \begin{cases} \varepsilon_{in} & \text{if} \quad \Delta p_{max_{i,j,k}} < \frac{p_{i,j,k}}{2} \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

$$\text{where} \quad \Delta p_{max_{i,j,k}} = \max\left(\Delta p_{i+}, \Delta p_{i-}, \Delta p_{j+}, \Delta p_{j-}, \Delta p_{k+}, \Delta p_{k-},\right)$$

$$\Delta p_{i+} = p_{i+1,j,k} - p_{i,j,k}, \quad \Delta p_{i-} = p_{i,j,k} - p_{i-1,j,k}$$

$$\Delta p_{j+} = p_{i,j+1,k} - p_{i,j,k}, \quad \Delta p_{j-} = p_{i,j,k} - p_{i,j-1,k}$$

$$\Delta p_{k+} = p_{i,j,k+1} - p_{i,j,k}, \quad \Delta p_{k-} = p_{i,j,k} - p_{i,j,k-1}$$

To further enhance stability, $\varepsilon_{i,j,k}$ is set to zero on the cut cells. This has the effect of creating first order accurate flux calculations in the cut cells.

Solid Surface Treatment

One final issue related to the inviscid fluxes is establishing the wall boundary conditions. In order to implement the surface tangency wall boundary conditions, the $\mathscr{F}_I$ flux in

equation (6) (or the non-dimensionalized for of equation (12)) yields

$$\mathcal{F}_I = \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \end{bmatrix} \tag{23}$$

since $\boldsymbol{v}_{wall} \cdot \boldsymbol{n}_{wall} = 0$, with p being found by satisfying the non-curved wall boundary condition $\frac{dp}{dn} = 0$.

Viscous Flux Calculations

The viscous flux calculations are split into two types, the simpler flow cell formulation and the more complicated solid surface cell formulation. The viscous flux formulations are simplified by the Cartesian nature of the control volumes, with more attention needing to be paid to the surface treatment.

Flow Cells

The viscous flux calculations of the flow cells are performed using standard second order finite difference approximations. The difference stencil is populated such that at refinement boundaries the differencing still appears as a uniform sized grid, which results in a less than second order accuracy for these regions. Page 58 provides more details on the stencil population. Since all of the flow cell faces are coordinate aligned, a large number of viscous terms do not need to be calculated in the $\mathcal{F}_V \cdot \boldsymbol{n}$ term from equations (5) and (6).

Solid Surface Treatment

The solid surface treatment of the viscous flux calculations requires the decomposition of the control volume velocities into surface oriented directions. To calculate the viscous fluxes for the surface face, a local coordinate system is defined such that η is normal to the surface and ξ and ζ are perpendicular to each other and are along the surface in order to form a right-handed orthogonal coordinate system (the actual directions of ξ and ζ are not important as will be shown later).

The transformation of the x-, y- and z-derivatives into ξ-, η- and ζ-derivatives is

$$\begin{aligned}
\frac{\partial}{\partial x} &= \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial y} &= \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial z} &= \frac{\partial \xi}{\partial z}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial z}\frac{\partial}{\partial \zeta}
\end{aligned} \tag{24}$$

For all quantities that do not vary on the surface (i.e. velocity, temperature in isothermal wall conditions and thin-layer Navier-Stokes approximations to temperature field) the $\frac{\partial}{\partial \xi}$ and $\frac{\partial}{\partial \zeta}$ terms are zero, and the transformation reduces to

$$\begin{aligned}
\frac{\partial}{\partial x} &= n_x \frac{\partial}{\partial \eta} \\
\frac{\partial}{\partial y} &= n_y \frac{\partial}{\partial \eta} \\
\frac{\partial}{\partial z} &= n_z \frac{\partial}{\partial \eta}
\end{aligned} \tag{25}$$

noting that $\frac{\partial \eta}{\partial x}$, $\frac{\partial \eta}{\partial y}$ and $\frac{\partial \eta}{\partial z}$ are just the slopes of the normal vector from the surface to the cell center: n_x, n_y and n_z.

To find the x-, y- and z-distances from the surface to the cell center, a standard formula can be used that finds the shortest distance between a surface and a point, see [74], to get the following values

$$
\begin{aligned}
n_x &= \frac{\boldsymbol{a}\cdot\boldsymbol{x}_c - d}{\boldsymbol{a}\cdot\boldsymbol{a}} a_0 \\
n_y &= \frac{\boldsymbol{a}\cdot\boldsymbol{x}_c - d}{\boldsymbol{a}\cdot\boldsymbol{a}} a_1 \\
n_z &= \frac{\boldsymbol{a}\cdot\boldsymbol{x}_c - d}{\boldsymbol{a}\cdot\boldsymbol{a}} a_2
\end{aligned}
\tag{26}
$$

where $\boldsymbol{a}\cdot\boldsymbol{x} - d = 0$ is the equation of the surface and $\boldsymbol{x}_c$ is the cell center. From equations (25) and (26), the viscous fluxes on the surface can be calculated.

Numerical Stencil Population

In order to calculate the inviscid and viscous fluxes, a numerical stencil must be constructed such that the necessary neighbor information can be determined. NASCART firsts determines the state vectors on the same mesh as the local cell and then performs a uniformly-spaced finite difference approximation to calculate the fluxes. With the possibility of mesh refinement in the grid, there are three grid configuration possible, a locally uniform grid, a local grid with fine neighbors and a local grid coarse neighbors.

The simplest case is that of a locally uniform grid, Figure 8. For this case no special treatment is required and the state of the neighboring control volumes, can be used as is. The label 'X' is used to denote the location of the needed state information.

For the case of the local grid having fine neighbors, Figure 9, the state information of the fine neighbor, labeled as 'o', is averaged together to create the required state information.

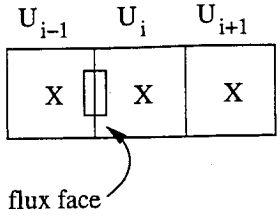


Figure 8: Uniform Stencil Population Example

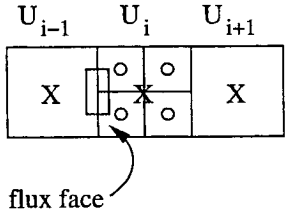


Figure 9: Fine Stencil Population Example

The final case is where the local grid has coarse neighbors, Figure 10. For this case, the state information for the coarse control volume is used as the state information at the desired locations. This reduces the local accuracy of the scheme, but also provides more dampening for any instabilities.

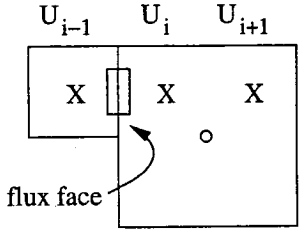


Figure 10: Coarse Stencil Population

Table 1 shows the stencil sizes for various schemes using this approach.

Table 1: Stencil Size for Each Face

Scheme	2D	3D
first order Euler	2	2
second order Euler	4	4
first order Navier-Stokes	6	10
second order Navier-Stokes	8	12

Time Integration

The time integration within NASCART is performed using a standard 2-stage Hancock integration scheme, with implementation details provided in reference [159]. Using the semi-discretized form of equation (5) results in the following

$$\boldsymbol{U}_{i,j,k}^{n+\frac{1}{2}} = \boldsymbol{U}_{i,j,k}^{n} - \frac{1}{2}\Delta t_{i,j,k}^{n} \iint_{CS_{i,j,k}} (\mathscr{F}_I^n - \mathscr{F}_V^n) \cdot \boldsymbol{n}\, dA \tag{27}$$

$$\boldsymbol{U}_{i,j,k}^{n+1} = \boldsymbol{U}_{i,j,k}^{n+\frac{1}{2}} - \Delta t_{i,j,k}^{n} \iint_{CS_{i,j,k}} \left(\mathscr{F}_I^{n+\frac{1}{2}} - \mathscr{F}_V^{n+\frac{1}{2}}\right) \cdot \boldsymbol{n}\, dA$$

where evaluation of the surface integrals will be discussed on page 61. Notice that the inviscid and viscous fluxes in the corrector steps are calculated using the state vectors generated from the predictor steps, and that local time-stepping can be employed if the steady-state solution is only desired.

Solution Adaption

The solution adaption methodology used in NASCART is similar to the velocity divergence approach discussed by Tu [160] where for each control volume, the velocity divergence is scaled by a characteristic length of the control volume to obtain a measure of the changing flow properties from cell to cell via

$$\tau_{d_{i,j,k}} = |\nabla \cdot v_{i,j,k}| l_{i,j,k}^{\frac{3}{2}} \tag{28}$$

where l is the cube-root of the cell volume.

Next the root-mean-square is calculated over the entire computational domain to obtain

a reference value, σ_d, using

$$\sigma_d = \frac{1}{N}\sum_{i=1}^{N} \tau_d^2 \tag{29}$$

Finally, cells are flagged for coarsening or refinement if the following conditions apply

$$\begin{aligned} \tau_{d_{i,j,k}} &< k_c \sigma_d \quad \text{coarsen} \\ \tau_{d_{i,j,k}} &> k_r \sigma_d \quad \text{refine} \end{aligned} \tag{30}$$

where k_c and k_r are threshold values for coarsening and refining, respectively.

Putting It All Together

Finally, the surface integrals in the Hancock time integration scheme (27) can be replaced with

$$\begin{aligned} \iint_{CS_{i,j,k}} (\mathscr{F}_I - \mathscr{F}_V)\cdot \boldsymbol{n}\, dA \approx\ & (\mathscr{F}_I - \mathscr{F}_V)_{i+1/2,j,k} A_{1_{i,j,k}} - (\mathscr{F}_I - \mathscr{F}_V)_{i-1/2,j,k} A_{2_{i,j,k}} \\ & + (\mathscr{F}_I - \mathscr{F}_V)_{i,j+1/2,k} A_{3_{i,j,k}} - (\mathscr{F}_I - \mathscr{F}_V)_{i,j-1/2,k} A_{4_{i,j,k}} \\ & + (\mathscr{F}_I - \mathscr{F}_V)_{i,j,k+1/2} A_{5_{i,j,k}} - (\mathscr{F}_I - \mathscr{F}_V)_{i,j,k-1/2} A_{6_{i,j,k}} \\ & + \left(\mathscr{F}_{I_{wall}} - \mathscr{F}_{V_{wall}}\right) A_{wall_{i,j,k}} \end{aligned} \tag{31}$$

where A_1 is the area of the xmax face, A_2 is the area of the xmin face, A_3 is the area of the ymax face, A_4 is the area of the ymin face, A_5 is the area of the zmax face, A_6 is the area of the zmin face and A_{wall} is the area of the wall face (if the cell has one). Combining all of the above results in a scheme that is second order accurate in time and between first and third order accurate in space.

CART3D

CART3D is an explicit, finite volume Cartesian grid solver of the three-dimensional Euler equations that has been validated in a number of flow conditions and configurations [3, 4, 22]. CART3D originated from the research of Melton et al. [105]. Improvements to the grid generation schemes, geometry representation and flow field refinement techniques were later performed by Melton et al. [104]. An overhaul of the flow solver to include multigridding, shared memory parallelization and CPU cache-based performance enhancements were performed by Aftosmis et al. [3].

Solver

The solver portion of CART3D, called flowCart, uses a face-based data structure for the spatial integration techniques. Within each control volume a piecewise linear distribution is used for the state variable reconstruction for the flux calculations to produce a second order scheme. A least-squares procedure provides the gradient estimations within each cell which is based on the solution of the normal equations of the local mass matrix. Flux quadrature is performed by a midpoint integration coupled with either a van Leer flux-vector splitting [9] or an approximate Riemann solver of Colella [42]. In order to suppress the oscillations associated with higher order schemes, flowCart uses either the minmod flux limiter [159] or Venkatakrishnan's flux limiter [169].

In handling the temporal discretization, flowCart employs a modified Runge-Kutta explicit time-stepping scheme. It supports an arbitrary number of Runge-Kutta stages with the

number of stages and the coefficients user configurable, with the van Leer 3-stage and van Leer 5-stage optimally dampened schemes [167] the typical schemes used to get second order and third order temporal accuracy, respectively.

To further improve the convergence characteristics, flowCart uses a Full Approximation Storage (FAS) multigrid scheme [26] based on the work of Jameson [75] to accelerate the convergence of the solver using both V- and W-cycles as well as Full Multigrid V-cycles. The intergrid transfer occurs by direct injection for the restriction phases and linear interpolation for the prolongation. A local block Jacobi preconditioning on each control volume is possible in order to further accelerate convergence. The combination of the upwind spatial discretization and the preconditioning results in rapid convergence for the FAS multigrid scheme.

Grid Creation and Partitioning

A major focus for CART3D was the issues related to grid creation and partitioning. Efforts were made to improve the performance characteristics of the grid generation procedures. Surface cells are constructed using techniques originating in the field of computer graphics in order to quickly and efficiently process surface intersections with the Cartesian grid. Additional techniques are employed in order to ensure the accurate representation of the surface geometry in the grid. Also, efforts are made to increase the solver performance by optimally ordering the control volumes as well as by finding acceptable distributions of the control volumes over the parallel nodes to achieve excellent load-balancing. The result

is a grid generation performance of $1x10^6$ cells/minute on a moderately powered desktop workstation in 1997 [3].

In the grid generation process, the flow cells are stored as the cell centroid and refinement level so that the complete geometry could easily be recreated with the additional information of the initial grid distribution. For the cut and split cells, they are handled as an arbitrarily shaped polyhedra with the centroid of the cell being stored as well as the surface triangulation of the cut surface. Additionally, each grid location is converted from a floating point representation to an integer based representation by using a 64-bit integer for storing all three coordinates. Thus each coordinate has to be represented in 21-bits, resulting in a maximum relative resolution of $2^{-21} \approx 4.8x10^{-7}$ in each coordinate direction. This integer based addressing allows for very fast geometry calculations during the grid generation process.

While the surface cells accounted for only $\mathcal{O}(N^2)$ cells, and the flow cells account for $\mathcal{O}(N^3)$ cells, special attention is paid to efficiently addressing the surface cells in order to optimize performance without sacrificing accuracy. By using the integer based coordinates, intersections of control volumes and surface triangles are determined using the bitwise "and" (&) and "or" (|) operators. The coordinates of each cell is relative to the cell that is being tested for the intersection. Each vertex in the triangle is given an index that corresponds to the cell that it is in. If any of the sides of the triangle intersect an edge of the cell, then the triangle is intersected. The intersection test for a cell is given in equation (32). This results in an extremely fast algorithm since equation (32) typically takes 3 CPU clock

cycles (one for each bitwise operation) compared to the many CPU clock cycles required for floating point arithmetic.

$$\text{if } \left(facecode_j \;\&\; \left(coord_{v_1} \mid coord_{v_2} \mid coord_{v_3} \right) \neq 0 \right) \text{ then intersect} \qquad (32)$$

Figure 11 shows a example of the intersection test configuration in two-dimensions. Thus for $\triangle_{tuv}$, the coordinates for the vertices are $coord_t = 0000$, $coord_u = 0000$ and $coord_v = 0100$. The $facecode$ parameter is the coordinate of the cell adjacent to each face, thus for the face that intersects with $\triangle_{tuv}$, the $facecode$ is 0100. Plugging these values into 32 shows that only $facecode$ of 0100 produces a non-zero result, and it is the only edge that intersects the triangle. More details on this technique can be found in reference [3].

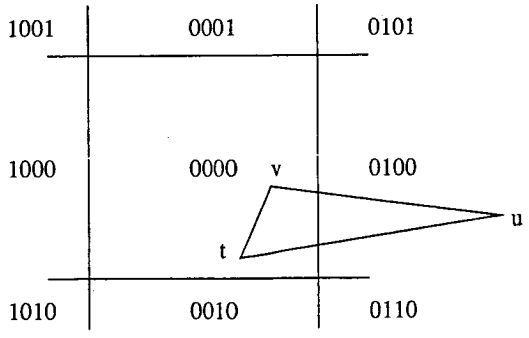


Figure 11: Example Surface Triangle Intersection with Cartesian Cell

In handling the surface triangles that intersect the Cartesian cells, the surface normals are used to determine if further grid refinement is needed. This is done by evaluating the change in angle of the surface normals for the surface triangles that intersect a cell, if the changes are above some threshold then more refinement will be required. Also, CART3D could use all of the intersecting surface triangles during the solution, or it could agglomerate

all of the surfaces into one area weighted average normal with a surface area equal to the sum of each triangles surface area. This functionality requires fewer calculations during the solution, while not adversely impacting the results. Figure 12 shows an example of the surface agglomeration. Notice that there are 3 sub-surfaces to the cut surface with normals n_1, n_2 and n_3 that get agglomerated into one surface normal n_{agg}, while the surface areas for all flow calculations and cell centroid determinations use the areas of the three original surfaces.

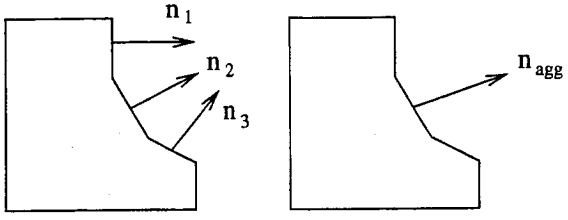


Figure 12: Example of Surface Agglomeration

Another grid technique that aids the solution process is the use of space-filling curves [144], or SFC, to generate the indexing of the cells. An effective SFC encourages better data locality for neighboring cells which results in better cache-based performance. The two orderings that Aftosmis et al. used were the Peano-Hilbert (or U-ordering) and the Morton (or N-ordering) schemes. Figures 13 and 14 show examples of Peano-Hilbert and Morton SFCs. Aftosmis et al. identified three characteristics that made these space-filling curve useful as a re-ordering technique [3]:

1. **Mapping** $\mathscr{R}^d \rightarrow \mathscr{H}$: Both ordering schemes provided unique mappings between the $\mathscr{R}^d$ physical space and a one-dimensional hyperspace, $\mathscr{H}$.

2. **Locality** : The U-ordering maintained adjacency of neighboring cells in the mapping between $\mathscr{R}^d$ and $\mathscr{H}$, while the N-ordering mostly maintained the adjacency of neighboring cells.

3. **Compactness** : The encoding and decoding of both orderings required only local information to generate the hyperspace indexing from the physical space coordinate and vice versa.

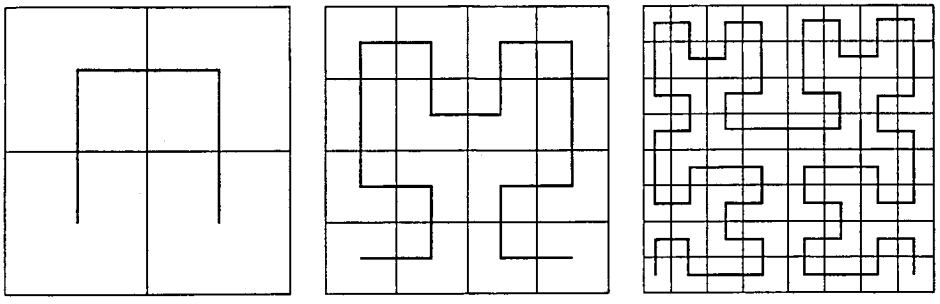


Figure 13: Example of Two-Dimensional Peano-Hilbert Curve

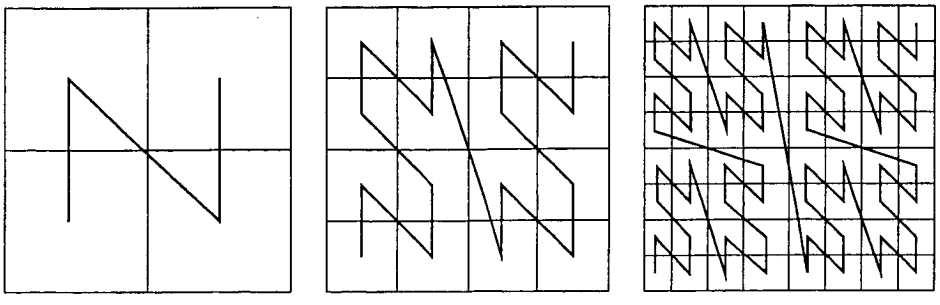


Figure 14: Example of Two-Dimensional Morton Curve

To generate the Peano-Hilbert curve in Figure 13, the template curve (the left curve) is recursively applied to every line segment such that starting and ending segments have the

template curve applied to the inside and the two corner segments have the template curve applied to the outside. The middle and right curves of Figure 13 show successive iterations of the Peano-Hilbert curve. Extending this to three dimensions is done in a similar manner using the template curve shown in Figure 15.

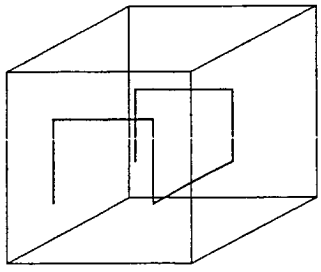


Figure 15: Example of Three-Dimensional Peano-Hilbert Curve

To generate the Morton curve in Figure 14, each quadrant is given a two-bit index representing the x and y location. Thus the lower left quadrant is 00, the upper left is 01, the lower right is 10 and the upper right is 11. The Morton curve is generated by traversing the quadrants in order of their two-bit index. Figure 14 shows three levels of iterations of the Morton curve. Extending this to three dimensions is done similarly to the two dimension case except that the octants are represented by a three-bit index representing the x, y and z locations as shown in Figure 16.

Using the space-filling curves as the ordering mechanism, Figure 17 shows an example mapping of a two-dimensional physical space domain with mixed levels of refinement to a one-dimensional hyperspace using the Peano-Hilbert ordering and the integer based cell location scheme.

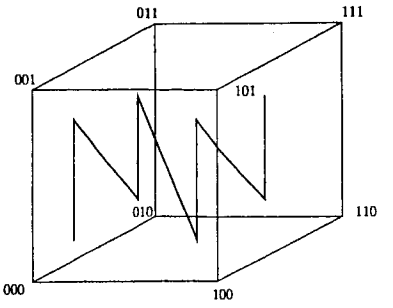


Figure 16: Example of Three-Dimensional Morton Curve

Handling of the domain decomposition for the parallelization of CART3D is done by simply splitting the SFC ordered cells evenly between the the processors, as shown in Figure 17. With the use of the SFC ordering, Berger et al. demonstrated that the resulting partitioning created roughly similar numbers of overlapping cells as did a perfectly uniform Cartesian mesh with the same number of cells [22]. In order to maintain favorable load-balancing characteristics, extra weighting is applied to cut and split cells in order to account for their higher computational cost. Thus partitions with a larger number of cut or split cells will have a lower overall number of cells.

CART3D uses a single pass scheme to create the grids for the multigrid solver. The procedure for coarsening the computational domains starts with the finest grid. This grid is then indexed using one of the SFCs mentioned above. At this point, the coarser grid levels of the multigrid solver can be created by a cell-by-cell traversal of the grid since the finest grid is already reordered. This results in the coarse grids retaining the SFC ordering. The grid coarsening procedure imposes a limit so that there is at most a refinement ratio of 2:1 on any grid. Thus some cells will not coarsen in the multigrid strategy until all of

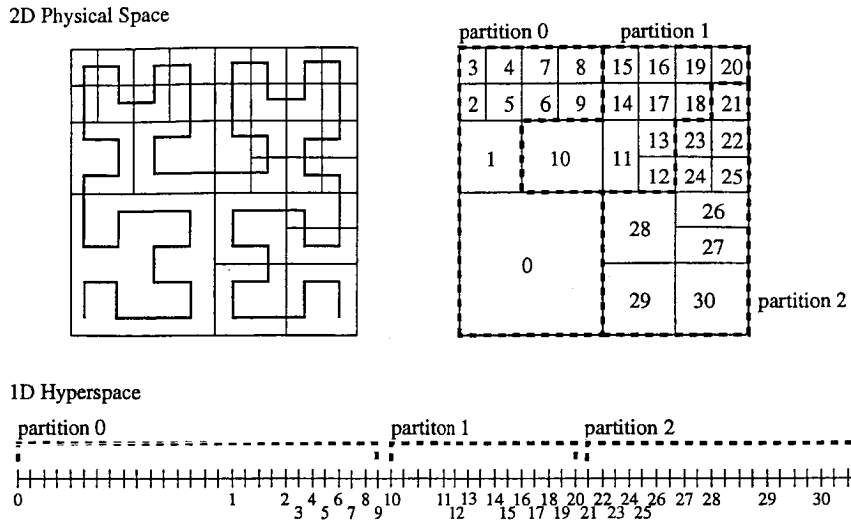


Figure 17: Two-Dimensional Mapping from Physical Space to Hyperspace

its neighbors have been coarsened. Finally, each grid is partitioned out to each processor using the SFC indexing in order to improve the overall load balancing characteristics of the solver. Figure 18 shows an example of one stage of coarsening around an arbitrary surface.

Special attention is paid to coarsening cut cells and split cells in order to handle the various coarsened grids that can result. Figures 19 and 20 show examples of the coarsening that can result around cut and split cells. Figure 19 shows 4 cut cells that coarsen to 2 cut cells, and Figure 20 shows 2 full cells and 4 split cells that coarsen to 2 cut cells. These are just two examples of the many variations that could occur during the coarsening process around cut and split cells.

The overall coarsening ratio, the ratio of fine cells to coarse cells in one coarsening

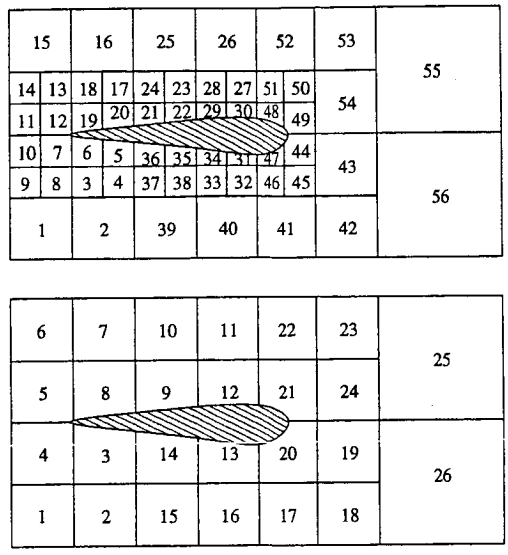


Figure 18: Grid Coarsening Around Arbitrary Surface

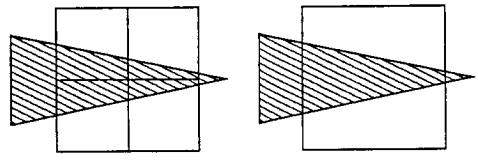


Figure 19: 4 Cut Cells Coarsen to 2 Cut Cells

step, for the CART3D coarsening procedures was shown to approach 7.25:1 for a variety of geometries [4] (noting that for a three-dimensional computational domain a perfect coarsening ratio would be 8:1). Also, this coarsening procedure was shown to be extremely fast, taking $\mathcal{O}(N \log N)$ steps to complete due to the quick-sort that occurs after the SFC indexing.

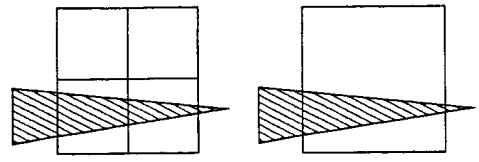


Figure 20: 2 Full Cells and 4 Split Cells Coarsen to 2 Cut Cells

Accuracy and Performance

CART3D was validated against both known analytical solutions as well as existing experimental data. Using the Supersonic Vortex model problem [5], Aftosmis et al. demonstrated a global order of accuracy of 1.88 [4] which compared favorably with other computational models. Additional validation was performed comparing results for an ONERA M6 wing in transonic flight conditions against experimental data with CART3D demonstrating all of the pertinent flow characteristics [4] as well as good agreement with pressure coefficient data.

CART3D uses OpenMP for its parallelization functionality with its parallelization performance showing excellent speedup results for up to 64 processors, with speedup figures of 28.4 and 52.3 for 32 and 64 processors, respectively [4, 22]. For all parallelization results, the residual histories for any number of CPU cases all matched to within machine accuracy due to the explicit nature of the time-stepping scheme and the lack of any iteration lagging in the updating of the overlapping cells [3].

The performance of the multigrid functionality of CART3D demonstrated a 5-times decrease in computation work to reduce the residual to machine zero for the test cases

mentioned above [4]. The parallelization performance of the multigrid functionality was not as good as the single-grid solutions since the coarser grids had smaller ratios of flow cells to overlapping cells.

CHAPTER III

SOLID BOUNDARY TREATMENT

The current schemes for calculating the solid-surface viscous boundary conditions all depend on calculating the wall shear stress and heat flux via numerical differences within the numerical solver in order to accurately calculate the numerical differences. This has been shown by Coirier [38] to produce extreme oscillations near the cut cells for even simple geometries due to the non-positivity of the stencils used in several viscous flux reconstruction techniques. In order to avoid these problems, the proposed approach uses special treatments for the solid boundary cells to provide a method of solving the Navier-Stokes equations on Cartesian grids. In addition, this scheme can be used to solve the Euler equations in order to eliminate the cut cells from the integration scheme and thus removing them from the time step restriction.

Existing Solid Boundary Treatment

The existing research into applying the Navier-Stokes equations to Cartesian grids, such as Frymier [59] and Coirier [38, 39], have utilized techniques to reconstruct the solid boundary fluxes in combination with the no slip wall boundary condition to model the solid boundary.

Frymier used simple extrapolation to obtain the wall pressure, with linear and quadratic curve fits for the velocity profiles to obtain the stresses. To model the heat flux at the wall, adiabatic wall boundary conditions were the only boundary conditions studied.

Like Frymier, Coirier used an extrapolation technique to obtain the wall pressure, but used a flux reconstruction technique to obtain the wall stresses using the wall centroid and the intersection points of the cell edge and the surface. For the wall heat flux boundary condition, an isothermal wall boundary condition was used with a one sided finite difference based derivative.

As was discussed in Chapter II, the Cartesian solver NASCART-GT originally determined the wall pressure by satisfying the normal momentum equation for a flat wall, $\frac{dp}{dn} = 0$, as well as a one sided finite difference formulation for the wall stresses and heat flux.

Unfortunately, all of these techniques produce unsatisfactory results when the resulting computational domain contains cut cells. As an example, while Coirier demonstrated excellent agreement with the Euler Cartesian grid solver, even simple flat plate Blasius configurations proved difficult to accurately capture when there were cut cells in the computational domain. Coirier's results for a Blasius flat plate configuration, grid shown in Figure 21, at $Re = 10{,}000$ with the plate at an angle of 30° with respect to the x-axis show large oscillations in the skin friction coefficient, shown here in Figure 22. This non-smoothness problem was observed in Frymier's work as well as in NASCART-GT, and it makes these solid surface boundary condition formulations of little use when general

bodies need to be modeled.

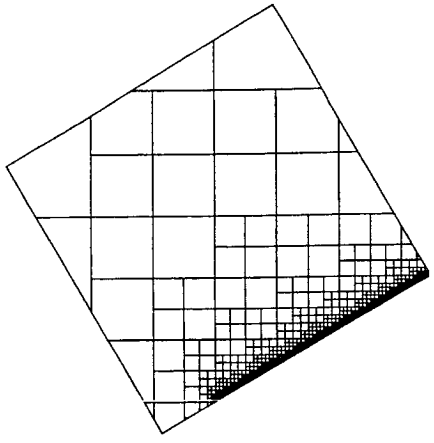


Figure 21: Grid from Coirier [38] for Rotated Blasius Flat Plate

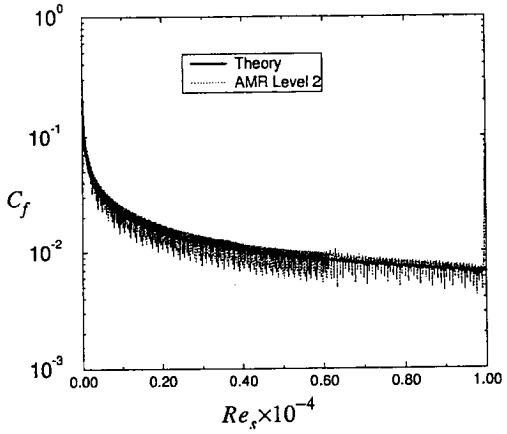


Figure 22: Skin Friction Results from Coirier [38] for Rotated Blasius Flat Plate

In addition to the non-smoothness problems associated with the existing solid boundary treatment, the cut cells generated by the solid surface intersecting with the Cartesian cells require very small time steps to maintain the CFL restriction needed to ensure the stability of the explicit time integration scheme. Thus to achieve solutions more efficiently, this time step restriction needs to be eliminated so that the minimum time step is set by the size of the smallest full cell.

New Solid Boundary Treatment

After reviewing the existing solid boundary treatments above, it becomes apparent that there needs to be a new treatment for the solid boundary condition that addresses the non-smoothness problems as well as the CFL restrictions associated with the cut cells. The new

approach presented addresses these problems by handling the solid body cells separately from the rest of the computational domain.

Basic Model Development

The problems associated with the non-smoothness of Navier-Stokes using Cartesian grids can be traced back to the non-positivity of the viscous flux stencil [38], thus a scheme for updating the state of the surface cells without using the viscous flux stencils needs to be used. One method of removing the dependence on the viscous flux stencils is to remove the surface cells from the finite volume formulation, while still using them for the flux reconstruction of its neighboring cells. The development of the state vectors for the surface cells can be obtained by satisfying the known criteria for the surface cells. Thus allowing the calculation of the majority of the control volumes in the computational domain to remain unchanged and can be treated as was discussed in Chapter II.

Reference State Determination

The formulation of the surface cell properties utilizes the state at a point normal to the surface which can be based on the surrounding cells, see figure 23. The state at point 'c' is constructed either directly from the state of the cell containing point 'c' (in this case labeled '5'), or by using a distance weighted interpolation of the of the surrounding cells (in this case cells '1' through '9'). The distance weighted interpolation places a restriction on the cells surrounding the surface cell such that all of the cells neighboring the reference cell and the reference cell itself must be at the same refinement level as the surface cell.

Using the state at point 'c', the state at the centroid of the surface cell, labeled '9', (or the wall location, labeled 'w') can be developed by using one-dimensional relationships along the line $\overline{Bw}$. The specifics of the state reconstruction depends on whether the flow is inviscid or viscous.

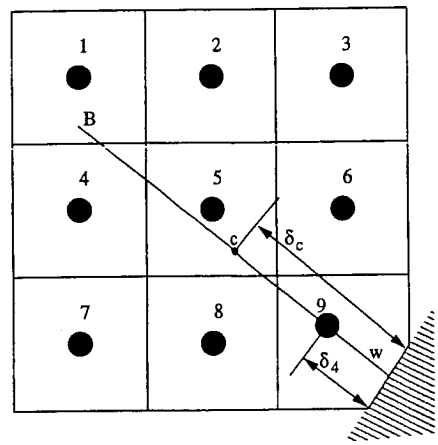


Figure 23: Example Configuration for Solid Boundary Treatment

Inviscid Formulation for Flat Wall

The inviscid formulation is separated into two cases, one if the flow at point 'c' is subsonic and another if it is supersonic.

Subsonic Case The surface cell velocity is first determined by an interpolation procedure along the line $\overline{Bw}$ from point 'c' to the wall utilizing the surface tangency wall boundary condition. The resulting relationship is

$$\boldsymbol{u}_9 = \boldsymbol{u}_C - \left(1 - \frac{\delta_9}{\delta_C}\right)(\boldsymbol{u}_C \cdot \boldsymbol{n})\boldsymbol{n} \tag{33}$$

where δ_c and δ_9 are the distances from point 'w' to points 'c' and '9', respectively. This has the effect of holding the tangential velocity constant and linearly decreasing the normal velocity to zero at the wall.

With the velocity determined at point '9', the temperature can be found by using the adiabatic relation

$$T_0 = T_c\left(1+\frac{\gamma-1}{2}M_c^2\right) \tag{34}$$
$$T_9 = T_0 - \frac{\gamma-1}{2C_p}U_9^2$$

and the pressure can be found by using the isentropic relation

$$p_9 = p_0\left(1+\frac{\gamma-1}{2}M_9^2\right)^{\frac{\gamma}{1-\gamma}} \tag{35}$$

This has the effect of correcting the thermodynamic properties for the velocity changes associated with the wall conditions.

Supersonic Case The supersonic case is split into two separate cases, one if the wall angle produces a shock and the other if it produces an expansion (or is parallel to the flow). If the wall produces a shock due to a positive wall angle, then the following standard

oblique shock relations are used, see [8] for the derivations

$$
\begin{aligned}
M_{n,c} &= M_c \sin\beta \\
\rho_9 &= \rho_c \frac{(\gamma+1)M_{n,c}^2}{(\gamma-1)M_{n,c}^2+2} \\
p_9 &= p_c \left[1+\frac{2\gamma}{\gamma+1}\left(M_{n,c}^2-1\right)\right] \\
M_{n,9}^2 &= \frac{M_{n,c}^2+\frac{2}{\gamma-1}}{\frac{2}{\gamma-1}M_{n,c}^2-1} \\
T_{n,9} &= T_c \frac{p_9}{p_c}\frac{\rho_c}{\rho_9} \\
M_9 &= M_{n,9}\csc(\beta-\theta) \\
\tan\theta &= 2\cot\beta\left[\frac{M_c^2\sin^2\beta-1}{M_c^2(\gamma+\cos 2\beta)+2}\right]
\end{aligned}
\tag{36}
$$

where β is the oblique shock angle and θ is the wall angle. One additional correction is to the velocity magnitude at '9'. The subsonic formulation from above is is used to calculate the velocity direction at '9', and the velocity magnitude from the oblique shock relations is used for the final velocity magnitude.

If the wall angle produces an expansion (or is parallel to the flow) then the same subsonic velocity relations are used to calculate the velocity vector. To calculate the thermodynamic properties, the standard Busemann surface pressure coefficient relation, see [25], is used to determine the pressure by

$$
\begin{aligned}
C_p &= \frac{2}{\sqrt{M_c^2-1}}\theta + \frac{(\gamma+1)M_c^2-4M_c^2+4}{2\left(M_c^2-1\right)^2}\theta^2 \\
p_9 &= p_c + \frac{\rho_c U_c^2}{2}C_p
\end{aligned}
\tag{37}
$$

where again θ is the wall angle. From the isentropic relations, the temperature at '9' is

calculated from

$$M_9^2 = \frac{2}{\gamma - 1}\left[\left(\frac{p_0}{p_9}\right)^{\frac{\gamma-1}{\gamma}} - 1\right] \tag{38}$$

$$T_9 = \frac{U_9^2}{\gamma R M_9^2}$$

Viscous Formulation for Flat Wall

As with the inviscid case, the viscous formulation is separated into two cases, one if the flow at point 'c' is subsonic and another if it is supersonic.

Subsonic Case The surface cell velocity is first determined by an interpolation procedure along the line $\overline{Bw}$ from point 'c' to the wall utilizing the no slip wall boundary condition. The resulting relationship is

$$\boldsymbol{u}_9 = \left[\boldsymbol{u}_C - \left(1 - \frac{\delta_9}{\delta_C}\right)(\boldsymbol{u}_C \cdot \boldsymbol{n})\boldsymbol{n}\right]\left(\frac{\delta_9}{\delta_C}\right) \tag{39}$$

where δ_c and δ_9 are the distances from point 'w' to points 'c' and '9', respectively. This has the effect of linearly decreasing the tangential velocity to zero and quadratically decreasing the normal velocity to zero at the wall.

Next, the pressure at point '9' can be determined by using the normal momentum equation for a flat wall to get

$$\frac{dp}{dn} = 0 \tag{40}$$

which when used in a first order forward finite difference approximation yields

$$p_9 = p_c \tag{41}$$

To close the thermodynamic system and enforce the final wall boundary condition, the temperature for the surface cell is determined. For an adiabatic wall boundary condition, a first order finite difference formulation for the wall heat flux yields the simple relation

$$T_9 = T_c \tag{42}$$

While for the isothermal case, a simple linear interpolation along BW, similar to the velocity formulation shown above, yields

$$T_9 = \frac{\delta_9}{\delta_c} T_c + \left(1 - \frac{\delta_9}{\delta_c}\right) T_w \tag{43}$$

Supersonic Case The supersonic case should be a pathological case since the wall cell must be in the boundary layer (thus subsonic), but it is applicable when the solution domain is initialized using the freestream values. If the wall angle produces a shock then the subsonic viscous velocity formulation is used to determine the velocity direction and the oblique shock relations are used to calculate the velocity magnitude and the thermodynamic conditions. Otherwise, the viscous subsonic formulations are used.

Curved Wall Model Development

While the basic model does address many of the problems that have been mentioned above, some deficiencies of the basic model have been addressed with the updated model. Specifically, utilizing the surface curvature to ease the grid refinement criteria around regions of high curvature, and utilizing the governing equations to develop the interpolation relationships. The surface curvature modification requires the governing equations

to be transformed into geodesic coordinates in order to incorporate the surface curvature terms. Appendix A provides the derivation details associated with the full Navier-Stokes equations, the boundary layer equations and the Euler equations in both two- and three-dimensions for geodesic coordinates.

Surface Curvature Determination

The geodesic coordinate directions, ξ, η and ζ, need to be defined for each surface cell so that the transformed governing equations can be used. Next, the necessary curvatures need to be calculated. Finally, the local velocity vectors need to be transformed from the Cartesian coordinate system to the geodesic coordinate system and back. The following sections provide the details for each of these steps.

Defining Geodesic Coordinate Directions In order to use the governing equations derived in Appendix A, the geodesic coordinate system for each panel must be determined. Recall that the ξ-and ζ-directions are along the surface, while the η-direction is normal to the surface. Further, recall that the surface is represented by a collection panels that can each be described by their unit normal vector, $\boldsymbol{n}$, and the location of the centroid panel, $\boldsymbol{x}_c$ in the following equation

$$\boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{x}_c) = 0 \tag{44}$$

Thus, the η-direction is simply the surface normal. The definition of the ξ-direction is the freestream velocity vector,$\boldsymbol{U}_\infty$, projected onto the surface. This is done so that the ξ-direction is the primary flow direction for most of the surface panels. For the panels that

are perpendicular to the flow direction (i.e. $\boldsymbol{n} \parallel \boldsymbol{U}_\infty$), then the ξ-direction is taken to be the direction of an edge of the panel ($\boldsymbol{e}_0$). The definition of the ζ-direction is such that it is normal to the other two directions to form a right-handed system. Thus, the three coordinate directions are defined as

$$\begin{aligned} \bar{\imath}_\xi &= \begin{cases} \frac{\boldsymbol{e}_0}{|\boldsymbol{e}_0|}, & \text{if } \boldsymbol{n} \parallel \boldsymbol{U}_\infty; \\ \frac{\boldsymbol{U}_\infty - (\boldsymbol{n}\cdot\boldsymbol{U}_\infty)\boldsymbol{n}}{|\boldsymbol{U}_\infty - (\boldsymbol{n}\cdot\boldsymbol{U}_\infty)\boldsymbol{n}|}, & \text{otherwise.} \end{cases} \\ \bar{\imath}_\eta &= \boldsymbol{n} \\ \bar{\imath}_\zeta &= \bar{\imath}_\xi \times \bar{\imath}_\eta \end{aligned} \tag{45}$$

Curvature Calculation Point Selection With the coordinate directions defined on the surface panels, the local curvatures can now be calculated. Figure 24 shows a typical three-dimensional surface configuration. For both the three dimensional Euler and boundary

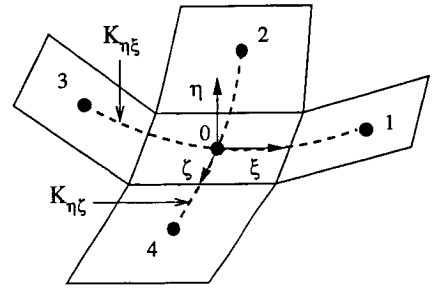


Figure 24: Example Surface for Curvature Calculation

layer geodesic formulations of the momentum equation in the η-direction, the required curvatures are $K_{\eta\xi}$ and $K_{\eta\zeta}$. These correspond to the curvature of the surface in the ξ- and ζ-directions, respectively (the arcs labeled $K_{\eta\xi}$ and $K_{\eta\zeta}$ in Figure 24).

In order to approximate the local surface curvature, the neighboring surface panels in the direction of the ξ- and ζ-coordinate axis are used to determine the local curvature by fitting a circular arc onto 3 points on the local surface panels. In figure 24, the calculation of the $K_{\eta\xi}$ curvature uses panels 0, 1 and 3 to build the arc, while the calculation of the $K_{\eta\zeta}$ curvature uses panels 0, 2 and 4.

For most cases, the neighboring panels in the positive and negative coordinate direction can be used to build the arc for the curvature calculations, however two special cases need to be addressed. The first is where the panel to be calculated is at a sharp edge, as shown in Figure 25. In this case, it would not be appropriate to use the panel on the other side of the edge in the calculation of the curvature because the curvature calculated would be too large. Instead the sharp edge itself is used. The second special case is when both neighboring panels form sharp edges, as shown in Figure 26. In this case, the same logic used in the sharp edge case discussed above is used for this case for the determination of the points to use in the calculation of the curvature. Thus, both directions use the edges in the curvature calculation, however, since all three points lie on the same plane, the curvature is zero. The determination of whether a corner is sharp is made by examining the angle between the normal vectors for the panels. If the angle between the normal vectors is $\pi/2$ or greater, then the two panels form a sharp edge.

Projecting Points onto Geodesic Coordinate System With the three points chosen to calculate the curvature from, the next step is to transform the problem into a two-dimensional

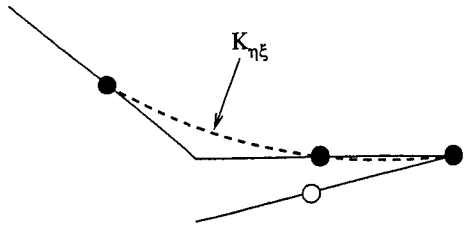


Figure 25: Single Sharp Edge Degenerate Surface

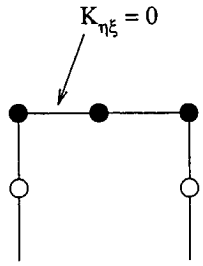


Figure 26: Double Sharp Edge Degenerate Surface

problem so that the circular arc can be found. This is done by constructing a local coordinate system centered at the center point (i.e. the point on the panel being evaluated) and projecting the vectors to the other two points onto the ξ- and ζ-coordinate direction vectors obtained above. Thus, for each point, ℓ, its local Cartesian coordinates, (x_ℓ, y_ℓ, z_ℓ) map to a geodesic coordinate, $(\xi_\ell, \eta_\ell, \zeta_\ell)$. If the $K_{\eta\xi}$ curvature is needed, then the ξ_ℓ and η_ℓ values are used, and if the $K_{\eta\zeta}$ curvature is needed, then the η_ℓ and ζ_ℓ values are used.

Curvature Determination The curvature for a panel on the surface given the three surface points projected onto the local geodesic coordinate system is found by substituting the three points, defined as (x_a, y_a), (x_b, y_b) and (x_c, y_c), into the following equations derived in Appendix B

$$
\begin{aligned}
R &= \pm \frac{\sqrt{\left[(x_a - x_b)^2 + (y_a - y_b)^2\right]\left[(x_a - x_c)^2 + (y_a - y_c)^2\right]\left[(x_c - x_b)^2 + (y_c - y_b)^2\right]}}{2\left[x_c(y_a - y_b) + x_b(y_c - y_a) + x_a(y_b - y_c)\right]} \\
x_0 &= \frac{(x_c^2 + y_c^2)(y_a - y_b) + (x_b^2 + y_b^2)(y_c - y_a) + (x_a^2 + y_a^2)(y_b - y_c)}{2\left[x_c(y_a - y_b) + x_b(y_c - y_a) + x_a(y_b - y_c)\right]} \\
y_0 &= -\frac{(x_c^2 + y_c^2)(x_a - x_b) + (x_b^2 + y_b^2)(x_c - x_a) + (x_a^2 + y_a^2)(x_b - x_c)}{2\left[x_c(y_a - y_b) + x_b(y_c - y_a) + x_a(y_b - y_c)\right]}
\end{aligned}
\tag{46}
$$

where R is the radius of curvature, and x_0 and y_0 are the locations of the center of the circle. There is an ambiguity in equation (46) associated with the sign of R. This can be resolved by examining the distance from the centroid of the cell associated with the panel that is being evaluated to the center of the arc. If this distance is larger than the circle radius, then the surface is convex and the appropriate sign is positive. Otherwise the surface is concave and the radius is taken to be negative. With this the surface curvature calculation is complete.

Normal Momentum Equations

The normal momentum equations are the source of the curvature corrections to the surface pressure values. For the inviscid formulation the three-dimensional curvature correction starts with the normal momentum equation in the geodesic coordinate system, developed in Appendix A and re-stated here

$$\frac{\partial p}{\partial \eta} = \rho \left[K_{\eta\xi} u_\xi^2 + K_{\eta\zeta} u_\zeta^2 \right] - \rho \left[\frac{\partial u_\eta}{\partial t} + \left(\frac{1}{h_\xi} \right) u_\xi \frac{\partial u_\eta}{\partial \xi} + u_\eta \frac{\partial u_\eta}{\partial \eta} + \left(\frac{1}{h_\zeta} \right) u_\zeta \frac{\partial u_\eta}{\partial \zeta} \right] \quad (47)$$

Applying equation (47) to the surface and utilizing the boundary conditions for the Euler flows (i.e. $u_\eta = 0$, $\frac{\partial u_\eta}{\partial t} = 0$, $\frac{\partial u_\eta}{\partial \xi} = 0$ and $\frac{\partial u_\eta}{\partial \zeta} = 0$) yields

$$\frac{\partial p}{\partial \eta} = \rho \left[K_{\eta\xi} u_\xi^2 + K_{\eta\zeta} u_\zeta^2 \right] \quad (48)$$

Notice that in equation (48) the sign of the curvatures ($K_{\eta\xi}$ and $K_{\eta\zeta}$) have a significance. Recall that as discussed above a sign was assigned to the curvature such that a positive curvature was generated by a convex surface, while a negative curvature was generated by

a concave surface. The sign of the curvature effects the direction of the pressure gradient. To adapt this to two dimensions, simply set the ζ-direction surface curvature to zero to get

$$\frac{\partial p}{\partial \eta} = K_{\eta\xi} \rho u_{\xi}^2 \tag{49}$$

Notice that this formulation is different from Wang and Sun [183] by a factor of -1, but their denominator for the radius, R, also differs by -1. Thus, curvatures that are positive for their system are negative for this system, and the resulting pressure gradient is the same sign.

The geodesic formulation of the boundary layer equations for the three-dimensional geodesic coordinate system yields the expression for the normal pressure gradient that will be required in this section. This equation, developed in Appendix A, is re-stated here

$$\frac{\partial p}{\partial \eta} = \rho \left[K_{\eta\xi} u_{\xi}^2 + K_{\eta\zeta} u_{\zeta}^2 \right] \tag{50}$$

Notice that this is valid throughout the boundary layer and not simply at the wall as was the case for the inviscid formulation. The same sign convention for the curvature is used here as for the inviscid formulation. A positive curvature is from a convex surface and a negative curvature is from a concave surface. A two-dimensional adaptation of this is found from setting the ζ-direction surface curvature to zero to get

$$\frac{\partial p}{\partial \eta} = \rho K_{\eta\xi} u_{\xi}^2 \tag{51}$$

Inviscid Wall Conditions for Curved Wall

The inviscid formulation is separated into two cases, one if the flow at point 'c' is subsonic and another if it is supersonic.

Subsonic Case The surface cell state calculation starts with the assumption that the normal velocity decreases linearly and that the magnitude of the velocity does not change between points 'c' and '9'. Further, it is assumed that the tangential velocity vector does not change directions with respect to the surface coordinates between points 'c' and '9'. The following expresses these criteria

$$\begin{aligned} \tan\lambda_c &= \frac{u_{\zeta,c}}{u_{\xi,c}} \\ u_{\eta,9} &= \frac{\delta_9}{\delta_c} u_{\eta,c} \\ u_{t,9} &= \sqrt{U_c^2 - u_{\eta,9}^2} \\ u_{\xi,9} &= u_{t,9}\cos\lambda_c \\ u_{\zeta,9} &= u_{t,9}\sin\lambda_c \end{aligned} \tag{52}$$

where u_ξ, u_η and u_ζ are the velocity components in the geodesic coordinate directions, u_t is the tangential velocity and λ is the angle made by the tangential velocity and the ξ-direction.

To develop the temperature relation, the adiabatic condition is used to get the following

$$\begin{aligned} T_0 &= T_c\left(1+\frac{\gamma-1}{2}M_c^2\right) \\ T_9 &= T_0 - \frac{\gamma-1}{2C_p}U_9^2 \end{aligned} \tag{53}$$

Notice that this has the effect of holding the temperature constant since the velocity magnitudes are the same between points 'c' and '9', thus T_w would also be equal to T_c.

With the temperature and velocity determined, the pressure relation can be developed by assuming a linear profile for the pressure curve along $\overline{Bw}$ and using equation (48) for

the slope of the pressure curve at the wall. To start, the equation for the pressure curve can be found to be

$$p = p_w + \delta \left.\frac{\partial p}{\partial \eta}\right|_w = p_w \left[1 + \frac{U_w^2}{RT_w}\left(K_{\eta\xi}\cos^2\lambda_c + K_{\eta\zeta}\sin^2\lambda_c\right)\delta\right] \tag{54}$$

where p is the pressure at a point δ distance away from the wall along $\overline{Bw}$. Since the conditions are 'c' as well as the temperature and velocity at the wall are known, the wall pressure can be solved for to get

$$p_w = \frac{RT_w}{RT_w + U_w^2 K_w \delta c} p_c \tag{55}$$

$$K_w = K_{\eta\xi}\cos^2\lambda_c + K_{\eta\zeta}\sin^2\lambda_c$$

where K_w is the combined curvature effects in the ξ and ζ directions. With the wall pressure found, the pressure at '9' can be found to be

$$p_9 = p_w\left(1 + \frac{U_w^2}{RT_w} K_w \delta_9\right) \tag{56}$$

and the boundary condition development is complete.

Supersonic Case The supersonic case is again split into two separate cases, one for a shock and the other for an expansion (or parallel flow). The shock case uses the oblique shock relations developed above to determine the velocity direction and thermodynamic conditions and the subsonic relations are used to determine the velocity direction. For the expansion or parallel flow case, the Busemann relations from above are used to determine the thermodynamic quantities while the subsonic relations are used to determine the velocity components.

Viscous Wall Conditions for Curved Wall

As with the inviscid case, the viscous formulation is separated into two cases, one if the flow at point 'c' is subsonic and another if it is supersonic.

Subsonic Case The subsonic viscous wall conditions again start with an assumption of the velocity profiles. As was the case for the inviscid wall conditions, the direction of the tangential velocity is assumed constant, i.e. $\lambda_c = \lambda_9$. Since there are only two conditions available to build a velocity profile around, the velocity at point 'c' and the no-slip boundary condition at the wall, the velocity profiles are limited to linear profiles defined as

$$\begin{aligned} u_\eta &= \frac{\delta}{\delta_c} u_{\eta,c} \\ u_t &= \frac{\delta}{\delta_c} u_{t,c} \end{aligned} \tag{57}$$

For the pressure boundary condition there are three conditions known, the pressure at point 'c' and $\frac{\partial p}{\partial \eta}$ at the wall and point 'c' from the boundary layer equations in geodesic coordinates derived in Appendix A. Applying the normal momentum equation of the boundary layer equation (50) to these conditions yields

$$\begin{aligned} \left.\frac{\partial p}{\partial \eta}\right|_w &= 0 \\ \left.\frac{\partial p}{\partial \eta}\right|_c &= \rho_c u_{t,c}^2 K_c \end{aligned} \tag{58}$$

where K_c is the same equation as the term K_w presented above, but applied at point 'c' instead of at the wall. With three conditions a quadratic profile can be used to describe the

pressure distribution throughout the boundary layer to get

$$p = \frac{1}{2}\left(\rho_c u_{t,c}^2 K_c \delta_c\right)\left[\left(\frac{\delta}{\delta_c}\right)^2 - 1\right] + p_c \tag{59}$$

The development of the final condition, temperature, utilizes the compressible boundary layer energy equation in geodesic coordinates, from Appendix A and restated here

$$\begin{aligned}\rho\frac{\partial H}{\partial t} + \rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi} + u_\eta\frac{\partial H}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right] \\ = \frac{\partial p}{\partial t} + \frac{\partial Q}{\partial t} + \frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\frac{\partial H}{\partial \eta} + \mu\left(1-\frac{1}{Pr}\right)\left(u_\xi\frac{\partial u_\xi}{\partial \eta} + u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right)\right]\end{aligned} \tag{60}$$

If steady state is assumed and the equation is applied to the wall (where $\boldsymbol{u} = \boldsymbol{0}$), then equation (60) becomes

$$\frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\frac{\partial H}{\partial \eta}\right]_w + \frac{\partial}{\partial \eta}\left[\mu\left(1-\frac{1}{Pr}\right)\left(u_\xi\frac{\partial u_\xi}{\partial \eta} + u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right)\right]_w = 0 \tag{61}$$

where all derivatives are taken at the wall. Converting this from stagnation enthalpy to temperature, $H = C_p T + U^2/2$, and recalling the constant specific heats assumption of NASCART-GT yields as well as the boundary layer assumptions that $u_t \gg u_\eta$, the linear velocity profile assumption yields

$$\left.\frac{\partial^2 T}{\partial \eta^2}\right|_w + \frac{Pr}{C_p}\left(\left.\frac{\partial u_t}{\partial \eta}\right|_w\right)^2 + \frac{1}{\mu_w}\left.\frac{\partial \mu}{\partial \eta}\right|_w\left.\frac{\partial T}{\partial \eta}\right|_w = 0 \tag{62}$$

Finally, if the assumption of constant viscosity at the wall is used then the boundary layer energy equation at the wall becomes

$$\left.\frac{\partial^2 T}{\partial \eta^2}\right|_w = -\frac{Pr}{C_p}\left(\left.\frac{\partial u_t}{\partial \eta}\right|_w\right)^2 \tag{63}$$

This condition along with the temperature at point 'c' provides two of the three conditions required for a quadratic curve fit. The third condition comes from the adiabatic or isothermal wall boundary condition.

For the adiabatic wall boundary condition, the third condition is $\frac{\partial T}{\partial \eta}\Big|_w = 0$ which results in the following equation for the temperature profile

$$T = -\frac{Pr}{2C_p} u_{t,c}^2 \left[\left(\frac{\delta}{\delta_c} \right)^2 - 1 \right] + T_c \tag{64}$$

For the isothermal wall boundary condition, the third condition is given by the wall temperature which results in the following equation for the temperature profile

$$T = -\frac{Pr}{2C_p} u_{t,c}^2 \left[\left(\frac{\delta}{\delta_c} \right)^2 - \left(\frac{\delta}{\delta_c} \right) \right] + (T_c - T_w) \frac{\delta}{\delta_c} + T_w \tag{65}$$

Special Surface Cell Treatment

The original objective of this alternative boundary condition treatment was to handle the arbitrarily small cut cells in a separate fashion so that they do not appear in the finite volume formulations (either for non-smoothness or time-step reasons). Thus, the primary focus is on cut cells that are much smaller than the flow cells that are at the same refinement level, see figure 27. Notice that the distance between point 'w1' and 'c1' is much larger

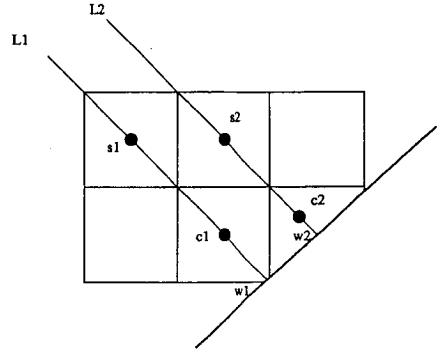


Figure 27: Large Cut Cell Example

than between 'w2' and 'c2'. This increased distance causes larger errors in the interpolation procedures. Since these are not the cells that are of primary importance, these cells should not be excluded from the finite volume integration. Including these cells in the finite volume formulation has the advantages of further reducing the cells that are not included in the finite volume integration and removing the cases where the largest interpolation errors will occur. There is a trade-off between the non-smoothness of the finite volume scheme and the accuracy of the interpolation procedures. If surface cells that are "too small" are included in the integration scheme, then the viscous formulation will become unstable and the inviscid

scheme will have its time-step greatly restricted, however if no surface cells are included in the integration scheme, then the interpolation inaccuracies will appear in these regions. In practice, the criteria used to determine the how small of a surface cell to include in the formulation is if the surface cells with volume 95% or more of the flow cell volume at the same refinement level are included in the integration.

State Reconstruction

Once the velocity, pressure and temperature are determined for the surface cell, the state vector can be reconstructed using the equation of state and the isentropic relation between internal energy and density to get the density, momentum and energy values from

$$\boldsymbol{U}_9 = \begin{bmatrix} \rho_9 \\ \rho_9 u_9 \\ \rho_9 v_9 \\ \rho_9 w_9 \\ E_9 \end{bmatrix} = \begin{bmatrix} \frac{p_9}{RT_9} \\ \rho_9 u_9 \\ \rho_9 v_9 \\ \rho_9 w_9 \\ \frac{p_9}{(\gamma-1)} + \frac{\rho_9(\boldsymbol{u}_9 \cdot \boldsymbol{u}_9)}{2} \end{bmatrix} \tag{66}$$

CHAPTER IV

PARALLELIZATION ENHANCEMENTS

Since the OpenMP code in flowCart (the flow solver in CART3D) was written following a domain decomposition strategy, each processor integrates only a sub-region of the entire domain, and then exchanges data at the boundaries of its subdomain. While this strategy is well suited for the the MPI parallelization of CART3D, there were several significant modifications that needed to be accomplished in order for the MPI port to be completed for the non-multigrid scheme. Most changes focused on handling the differences in the OpenMP and MPI paradigm, such as ensuring all processes receive the results of serial tasks and removing all dependencies on shared memory structures. All of these modifications were made such that the temporary memory requirements did not drastically increase with the storing of large amounts of configuration data. This chapter will discuss some of the more important changes that needed to be accomplished.

Initialization Information Distribution

One of the key differences between the OpenMP parallelization and the MPI parallelization is how the parallelization is accomplished. For OpenMP, threads are spawned for the parallelized regions of the code, leaving the rest of the code to be executed by a single instance

of the application. All data that exists for the serial portions of the code is automatically available for the parallel threads. For MPI, everything is executed as parallel processes, so any serial section must be delegated to one process while the others wait. Any data that needs to be available to all processes must be explicitly passed to all processes since MPI does not guarantee any data (including command line arguments) will be available to all processes.

The initialization process in the OpenMP version of flowCart (flowCart-OpenMP) consisted of parsing the command line arguments to get any initial configuration information, reading in the configuration file, and finally re-parsing the command line arguments for any configuration information that overrides the configuration file settings. All of this initialization was performed serially, and was followed by packing the configuration information into global data structures. For the MPI version of flowCart (flowCart-MPI), this needed to be changed such that the root process (the only process guaranteed to have access to the command line arguments and configuration file) performed all of the serial tasks from flowCart-OpenMP and then distributed the configuration information to the rest of the processes, via the `MPI_Bcast` function.

Grid Information Distribution

Once the configuration information was distributed to all of the MPI processes, all of the grid information needed to be distributed. This was done in flowCart-OpenMP in a section of serial code using two passes through the grid data file, with the first pass determining the

grid sizes for each process for appropriate load balancing and the second pass distributing the grids. As before, this was delegated to the root process. The first pass required little changes except for some extra internal buffers for the root process to store the grid sizes for each process. At the end of first pass, the root process distributes the grid sizes using the MPI call `MPI_Send`, and each process receives their grid size using the MPI call `MPI_Recv`, which is the followed by the allocation of the memory for the grid data by each process.

The second pass through the grid data file (where the grids are actually distributed to each process) required more attention associated with the exchange of data between grids. In addition to reading and distributing the grids, the information required to map partitions that share one or more faces is also constructed. Figure 28 shows a simple example of two partitions and the overlapping cells that each partition uses to store information about its neighbor.

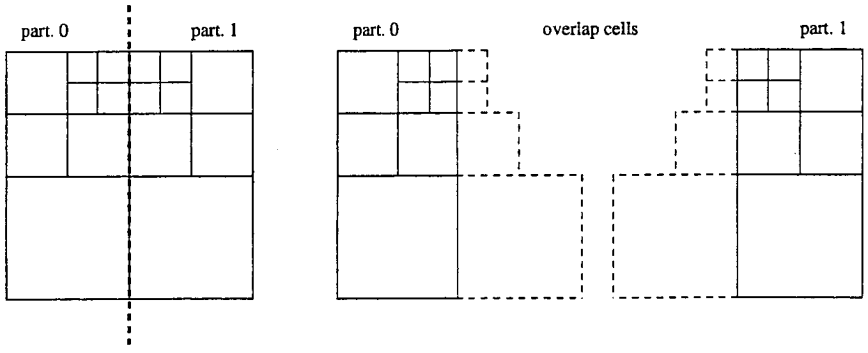


Figure 28: Overlapping Cell Configuration for flowCart

The indexing scheme that was used in the OpenMP parallelization to map the boundary overlap control volumes for each process to the flow volumes in another process needed to

be changed. It was setup for each grid to know where its overlapping cells mapped and then retrieve that data when needed. Thus for Figure 29, Table 2 shows the indexing scheme that flowCart-OpenMP would have used. Under this indexing scheme, when partition 0 needed to update overlap cell 1, corresponding to (0,1) in Table 2, the information was retrieved from partition 1, cell 1, corresponding to (1,1), and partition 0 only needed to store the integer pair (1,1) in order to update overlap cell 1. While this scheme worked well for flowCart-OpenMP, it would be very inefficient to try to implement this using MPI due to its strong dependence on direct access to physical memory locations.

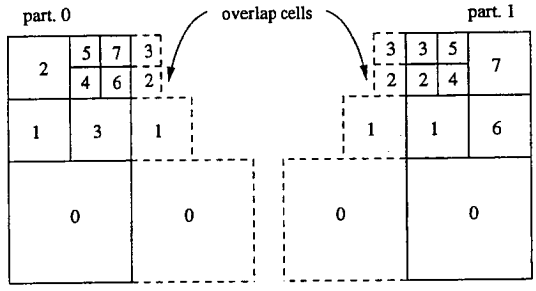


Figure 29: Overlapping Cell Indexing for flowCart

Since MPI is a message based communication scheme, emulating the updating of information for the example above would require too much bi-directional communication between processes. For partition 0 to update overlap cell 1, a message would first need to be sent to process 1 requesting the data from cell 1, then process 1 would need to send a message back to process 0 with the data. While this scheme could be improved by collecting all of the requests for data going to each process and performing fewer, larger requests and sends, this would still result in unnecessary overhead.

Table 2: Original Overlapping Cell Indexing for flowCart-OpenMP

Overlap Cell (part.,index)	Internal Cell (part.,index)	Stored Data (part.,index)
(0,0)	(1,0)	(1,0)
(0,1)	(1,1)	(1,1)
(0,2)	(1,2)	(1,2)
(0,3)	(1,3)	(1,3)
(1,0)	(0,0)	(0,0)
(1,1)	(0,3)	(0,3)
(1,2)	(0,6)	(0,6)
(1,3)	(0,7)	(0,7)

A more direct indexing scheme is to have each partition keep track of its cells that are needed by a particular process. For Figure 29, this would result in Table 3. Now overlap cell 1 in partition 0 is updated by partition 1 sending cell 1 to partition 0, and partition 1 only needs to store the index of its cell that needs to be sent, the partition to send the data, and the overlap cell index. Using this scheme, the exchange of data between partitions occurs in a uni-directional communication.

In addition to the overlap cell indexing change, significant efforts were made in order to not drastically increase the transient memory requirement on the root process in order to build all of the overlapping information. While there was an increase in the internal data structures required for the grid distribution process, the increase was negligible and had no overall impact on the memory usage.

State and Gradient Exchanges

Table 3: New Overlapping Cell Indexing for flowCart-OpenMP

Internal Cell (part.,index)	Overlap Cell (part.,index)	Stored Data (index,overlap part., overlap index)
(1,0)	(0,0)	(0,0,0)
(1,1)	(0,1)	(1,0,1)
(1,2)	(0,2)	(2,0,2)
(1,3)	(0,3)	(3,0,3)
(0,0)	(1,0)	(0,1,0)
(0,3)	(1,1)	(3,1,1)
(0,6)	(1,2)	(6,1,2)
(0,7)	(1,3)	(7,1,3)

In order for the solver to advance in time, the state and gradient information for the overlapping cells mentioned above needed to be exchanged using MPI calls instead of the current OpenMP functionality. This was easily accomplished by using the new overlap cell indexing scheme. Each process now loops over all of its cells that are overlap cells for other processes and packs the state (and later the gradient) data into message buffers (one for each process that is to receive data). Once the buffers are packed, they are sent using the non-blocking MPI send function `MPI_Isend`. This allows each process the ability to send all of its data so that it can be ready to receive its overlap cell data from other processes, using the `MPI_Recv` function. If the blocking form of the send function `MPI_Send` were used, then it is easy to see that dead-lock conditions could easily arise. Take a simple two process parallel exchange where the two processes both call `MPI_Send`, they will both be stuck waiting because the call will not return until the receiving process receives the data, but the receiving process is stuck waiting for its own send to complete. Thus, dead-lock

occurs.

While the use of the non-blocking send solves the dead-lock condition, using it as described above does introduce a possible problem. Having all processes sending their data at the same time can cause the memory connection bandwidth to become saturated, but in practice this appears to have no adverse performance effects. For severely bandwidth limited architectures, it is conceivable to create a communication scheduling algorithm so that each process would either send, receive, or wait for each step in the schedule. This schedule could be optimized to minimize the number of steps in the schedule or to minimize the total elapsed time in the schedule. However, since bandwidth saturation has not become an issue, these schemes will not be studied further.

Solution Reporting Mechanisms

Two changes were needed to be made to the solution reporting mechanisms in order to complete flowCart-MPI. The first change was to the residual calculations that occur after each solution iteration has been performed. For the residual calculations, there are two residuals that are calculated, the L1 and infinity norms of the density values. Each process continues to calculate its local residuals as before, but an additional step is added. At the end of each residual calculation, each process uses the MPI function `MPI_Allreduce` in order to determine the global residuals. The `MPI_Allreduce` function performs a traditional gather-scatter operation [123]. A gather-scatter operation is a communication operation that collects and processes information from a number of sources and distributes the results

to all processes that supplied the information. For the L1 norm, the operation specified to the MPI function is the sum operation (using `MPI_SUM`), while for the infinity norm, the operation specified is the max operation (using `MPI_MAX`).

The second change that needed to occur was to the extraction of cutting planes and surfaces that occurred during post-processing. As was the case for the residual calculations, each process performs its extraction calculations as before, but an additional step is added. After each process has performed its own extraction calculations and has created its own portion of the resulting cutting plane or surface, the root process cycles through all of the other processes and collects the plane or surface information and writes the data out. This data is not stored by the root node since doing so could cause a significant additional memory requirement on the root node, especially if the cut plane or surface is larger than the available memory. Thus, common sections between processes (i.e. overlap cells or shared faces) cannot be eliminated as they would in flowCart-OpenMP. This represents the only performance characteristic difference between flowCart-OpenMP and flowCart-MPI and in general is only a small portion of the overall extracted cutting plane or surface, ($< 0.1\%$).

CHAPTER V

SOLID BOUNDARY RESULTS

After implementing the modifications to NASCART-GT presented in Chapter III, tests were performed to determine the improvements made in the ability to model the solid boundaries in both inviscid and viscous flows in Cartesian grid formulations. These new solid boundary treatments remove the non-smoothness seen in the traditional viscous flux reconstruction techniques as well as the time-step limitations present in all current Cartesian solvers that include the surface cells in the integration procedure. This chapter presents a series of test cases that demonstrates the ability of NASCART-GT to handle a variety of inviscid and viscous flows.

Primitive Geometry Flows

The first set of cases are primitive geometry flows that have well studied solutions, either analytically or computationally, which can be used as a first stage of validation. To validate the inviscid wall boundary conditions, an incompressible cylinder flow and a compressible cylinder flow are studied. To validate the viscous wall boundary conditions, an incompressible flat plate flow and a supersonic flat plate flow are studied.

Incompressible Inviscid Cylinder Flow

The incompressible inviscid cylinder test case is a circular cylinder with a radius of 0.5, or a curvature of 2.0, in a $M_\infty = 0.1$ freestream flow. The computational boundaries are 10.5 diameters ahead and behind the cylinder and 10.5 diameters above and below the cylinder. The finest level of cells were ensured to be 0.5 diameters around the cylinder. Solutions are presented on two grids, one using a coarse grid of 84x84 root grid dimensions with 4 levels of refinement for a total of 10,056 cells and a fine grid with 5 levels of refinement for a total of 18,216 cells. Figures 30 and 31 show the coarse and fine grids, respectively. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure.

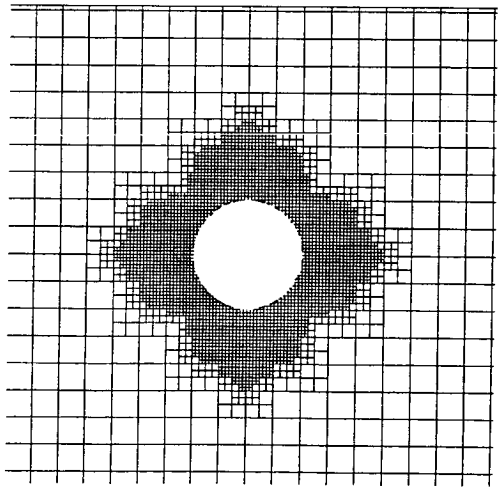


Figure 30: Coarse Computational Domain for Incompressible Cylinder Flow

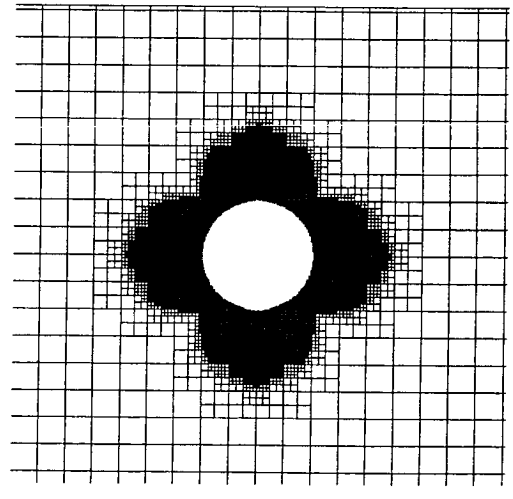


Figure 31: Refined Computational Domain for Incompressible Cylinder Flow

The cylinder curvature, calculated by NASCART-GT using the methods described in Chapter III, is within 0.1% of the true value 2 for the cylinder.

To assess the accuracy of the surface boundary conditions, the surface pressure is compared to the pressure obtained the incompressible potential flow solution

$$p(\theta) = p_\infty + \frac{1}{2}\rho_\infty U_\infty^2 \left(1 - 4\sin^2\theta\right) \tag{67}$$

where $\theta = 0°$ is the leading edge of the cylinder, $\theta = 90°$ is the pressure minimum on the upper half of the cylinder and $\theta = 180°$ is the trailing edge of the cylinder. Figure 32 shows a comparison between the flat wall and curved wall boundary conditions for the 1st order solution on the coarse grid. Both conditions accurately capture the front stagnation pressure and under predict the rear stagnation pressure. The rear stagnation pressure under predictions are most likely due to the numerical dissipation associated with the computational schemes employed. The curved wall boundary condition does a better job of capturing the pressure minimum with a 6.5% relative error compared to a 9.1% relative error for the flat wall boundary condition. Table 4 shows the front stagnation point, minimum pressure point and rear stagnation point pressure values for the flat wall and curved wall boundary conditions compared to the theoretical incompressible solution.

Table 4: Incompressible Cylinder Surface Pressure Values for 1st Order Solution

	flat p/p_∞	curved p/p_∞	exact p/p_∞
front stag.	1.0067	1.0067	1.0070
p_{min}	0.9879	0.9854	0.9790
rear stag.	0.9988	0.9988	1.0070

Table 5: Incompressible Cylinder Surface Pressure Values for 3rd Order Solution

	flat p/p_∞	curved p/p_∞	exact p/p_∞
front stag.	1.0076	1.0057	1.0070
p_{min}	0.9828	0.9791	0.9790
rear stag.	1.0017	1.0047	1.0070

The curved wall boundary condition solution has a C_l of 0.0000 and a C_d of 0.8166 while the flat wall boundary condition solution has a C_l of 0.0000 and a C_d of 0.8608. The

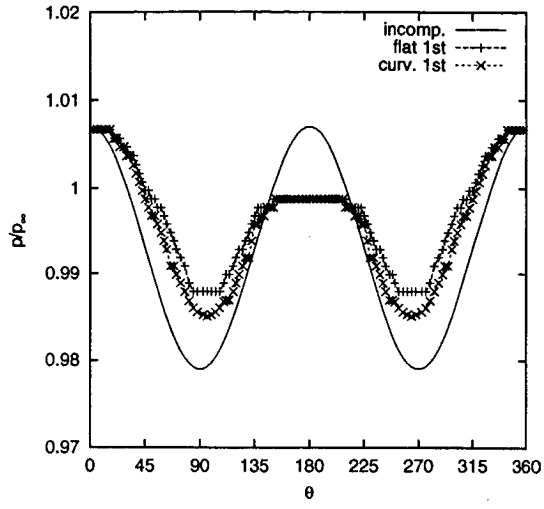


Figure 32: Incompressible Cylinder Surface Pressure 1st Order Solution with Interpolated Reference Points

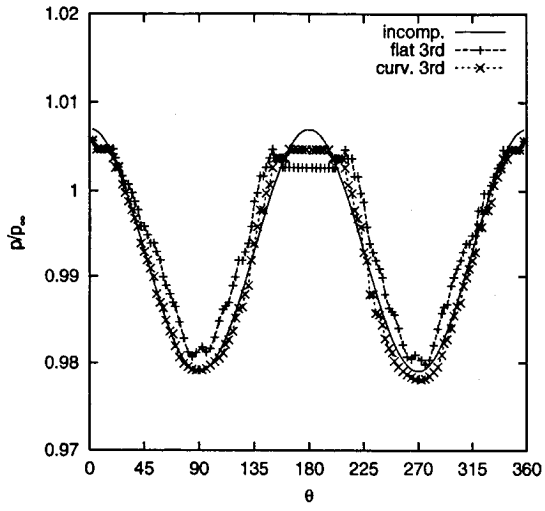


Figure 33: Incompressible Cylinder Surface Pressure 3rd Order Solution with Interpolated Reference Points

non-zero drag calculations are a result of the separation caused by the numerical dissipation discussed above. Table 7 shows the lift and drag coefficients for this case along with the other cases for the incompressible cylinder.

Figure 33 shows the results for the same configuration as above except using the 3rd order solver. For this case both solutions slightly under-predict the front stagnation pressure and under predict the rear stagnation pressure. Again, the numerical dissipation causes a separation region in the rear of the cylinder, but the separation point is moved much further back compared to the first order solution. The separation point for the first order solution is at $\theta \approx 156.^\circ$ while for the third order solution it is at $\theta \approx 167^\circ$. While both boundary condition schemes are better able to capture the pressure minimum using the 3rd order scheme, the curved wall boundary condition is again much better with a -0.01% relative error compared to a 0.2% relative error for the flat wall boundary condition.

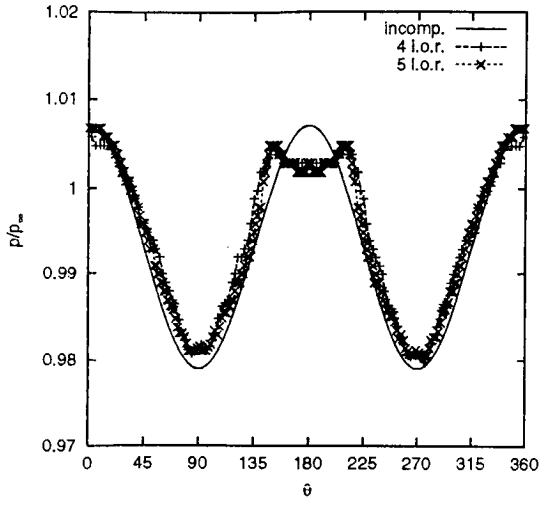


Figure 34: Fine Grid Incompressible Cylinder Surface Pressure Flat Wall with Interpolated Reference Points

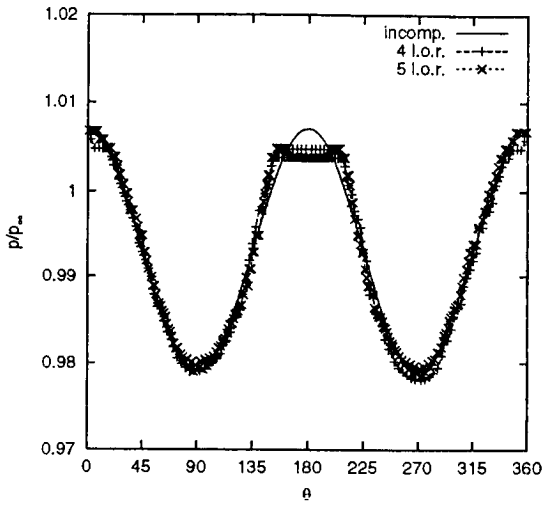


Figure 35: Fine Grid Incompressible Cylinder Surface Pressure Curved Wall with Interpolated Reference Points

Table 6: Incompressible Cylinder Surface Pressure Values for Fine Grid Solution

	flat p/p_∞	curved p/p_∞	exact p/p_∞
front stag.	1.0067	1.0067	1.0070
p_{min}	0.9815	0.9796	0.9790
rear stag.	1.0027	1.0037	1.0070

Table 6 shows the front stagnation point, minimum pressure point and rear stagnation point pressure values for the flat wall and curved wall boundary conditions compared to the theoretical incompressible solution. From table 7, the curved wall boundary condition solution using the third order solver has a C_l of -0.1001 and a C_d of -0.05578 while the flat wall boundary condition solution has a C_l of -0.00987 and a C_d of 0.2250. Again, the non-zero drag calculations are due to the separation caused by the numerical dissipation.

A comparison of the coarse grid and fine grid solutions are given in figures 34 and 35. Both figures show slight improvements to the surface pressure values around the entire surface. Table 6 shows the front stagnation pressure, pressure minimum and rear stagnation pressure results for both cases. For the fine grid, the curved wall boundary condition solution has a C_l of -0.0423 and a C_d of 0.1037 while the flat wall boundary condition solution has a C_l of -0.0409 and a C_d of 0.1134, see table 7.

Table 7: Incompressible Cylinder Lift and Drag Results

	first order coarse		third order coarse		third order fine	
	flat wall	curved wall	flat wall	curved wall	flat wall	curved wall
C_l	0.0000	0.0000	-0.00987	-0.1001	-0.0409	-0.0423
C_d	0.8608	0.8166	0.2250	-0.05578	0.1134	0.1037

Finally, figure 36 shows the grid convergence for the front stagnation point pressure error for the fine grid solution, the coarse grid solution, and one coarser grid not shown. The x-axis of this figure corresponds to the number of cells along the cylinder diameter in a coordinate direction. For coarser grids the order of accuracy is not quite second order, with the actual order of 1.31, while for the finer grids the order of accuracy is just above second order, with the actual order of 2.15.

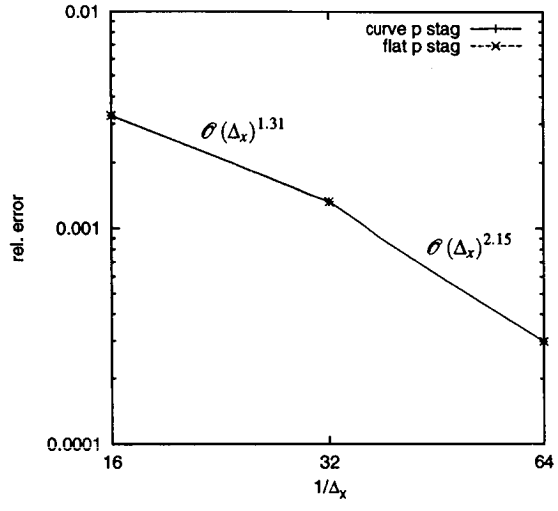


Figure 36: Incompressible Cylinder Grid Convergence with Interpolated Reference Points

Compressible Inviscid Cylinder Flow

The compressible inviscid cylinder test case is a circular cylinder with a radius of 0.5, or a curvature of 2.0, in a $M_\infty = 0.38$ freestream flow. The computational boundaries are 10 diameters ahead and behind the cylinder and 10 diameters above and below the cylinder. The finest level of cells were ensured to be 0.5 diameters around the cylinder. Solutions are presented on two grids, one using a coarse grid of 42x42 root grid dimensions with 5 levels of refinement for a total of 4884 cells and a refined grid with 6 levels of refinement for a total of 13,052 cells. Figures 37 and 38 show the coarse and fine grids, respectively. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure. Comparisons are made with results from Dadone and Grossman[44] for their structured grid solutions on a 128x32 (4096) cell domain, both with and without curvature corrections.

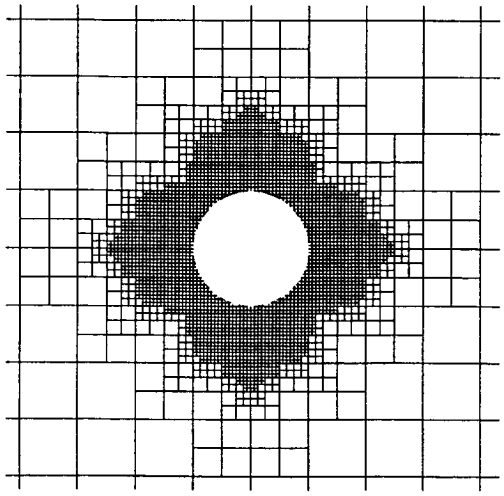


Figure 37: Original Computational Domain for Compressible Cylinder Flow

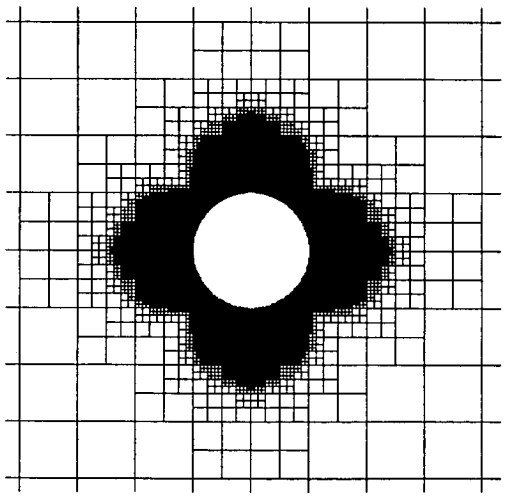


Figure 38: Fine Computational Domain for Compressible Cylinder Flow

Table 8 shows a comparison between the pressure at the front and rear stagnation locations and the pressure minimum location values for the flat wall and curved wall boundary conditions with the results from Dadone and Grossman. For the coarse grid solution, both boundary conditions are very close to the reference results for the front stagnation point. For the rear stagnation point, the same numerical dissipation effects discussed previously are apparent here with little difference between the two results. At the pressure minimum, the curved wall solution is significantly better with a relative error of 0.9% compared to 9.2% for the flat wall boundary condition solution. The curved wall boundary condition solution has a C_l of -5.857×10^{-4} and a C_d of 0.01905 while the flat wall boundary condition solution has a C_l of -5.760×10^{-3} and a C_d of 0.2221. The curved wall boundary condition significantly improved the lift and drag coefficients as well. Table 9 shows the lift and drag coefficients for this case along with the fine grid case for comparisons.

Table 8: Compressible Cylinder Surface Pressure Values

	Flat Wall		Curved Wall		
	coarse	fine	coarse	fine	Dadone
	p/p_0	p/p_0	p/p_0	p/p_0	p/p_0
front stag.	1.001	1.005	0.999	1.004	1.001
p_{min}	0.630	0.616	0.582	0.578	0.577
rear stag.	0.943	0.962	0.953	0.966	1.001

A comparison between the pressure values for the original and curvature boundary conditions for the fine grid from table 8 shows that the curved wall solution changed much less than the flat wall solution, with the flat wall solution substantially improving. The most

pronounced improvement is in the pressure minimum value with a relative error of 6.8% (compared to 9.2% for the coarse grid). The curved wall pressure minimum value improved to a relative error of 0.2% (compared to 0.9% for the coarse grid). The front stagnation point is slightly off (around 0.4%) and the rear stagnation point has improved, but is still showing the effects of numerical diffusion. The curved wall boundary condition solution has a C_l of -2.050×10^{-3} and a C_d of 0.0644 while the flat wall boundary condition solution has a C_l of -6.932×10^{-3} and a C_d of 0.1803. Again, the curved wall boundary condition significantly improved the lift and drag coefficients compared to the flat wall boundary condition, while both fine grid solutions result in a slight increase in lift and drag coefficients.

Table 9: Compressible Cylinder Lift and Drag Results

	third order coarse		third order fine	
	flat wall	curved wall	flat wall	curved wall
C_l	-5.760×10^{-3}	-5.857×10^{-4}	-6.932×10^{-3}	-2.050×10^{-3}
C_d	0.2221	0.01905	0.1803	0.06444

Finally, figures 39 and 40 show the Mach contours for the solutions on the fine grid with the flat wall boundary condition and the curved wall boundary condition, respectively. Figures 41 and 42 show the Mach contours from Dadone and Grossman for their flat wall boundary condition and curvature corrected boundary condition, respectively. Both sets of figures have $\Delta M = 0.1$ for the contours. Comparing the reference figures to the figures from NASCART-GT shows that both NASCART-GT boundary conditions perform quite well at capturing the flow features everywhere except near the rear stagnation point. Similar stagnation pressure losses that are present in the NASCART-GT cases can be seen in other results from Dadone and Grossman for less accurate wall boundary conditions.

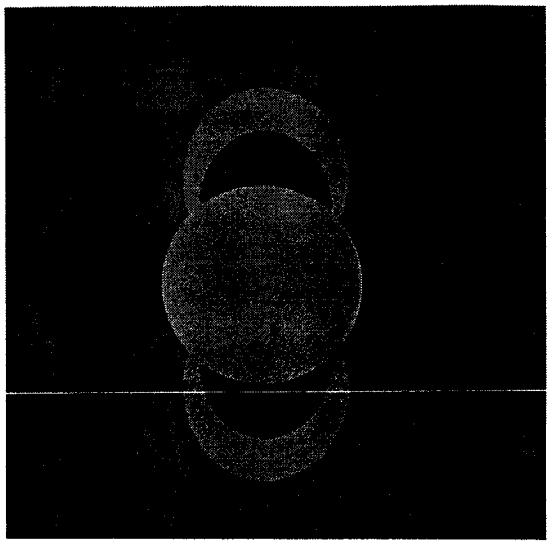


Figure 39: Mach Contours for Compressible Cylinder Flow Flat Wall with Interpolated Reference Points

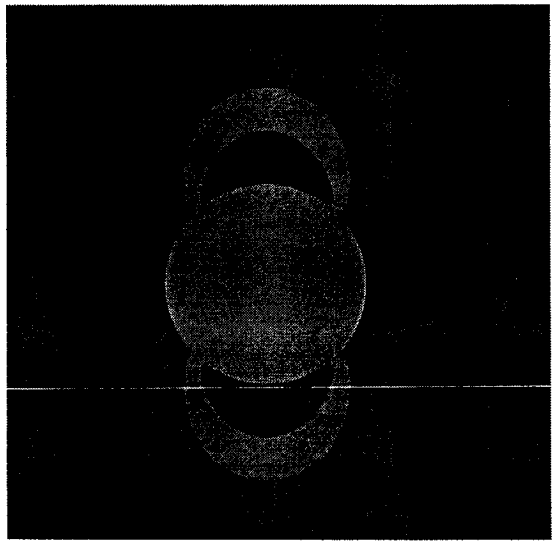


Figure 40: Mach Contours for Compressible Cylinder Flow Curved Wall with Interpolated Reference Points

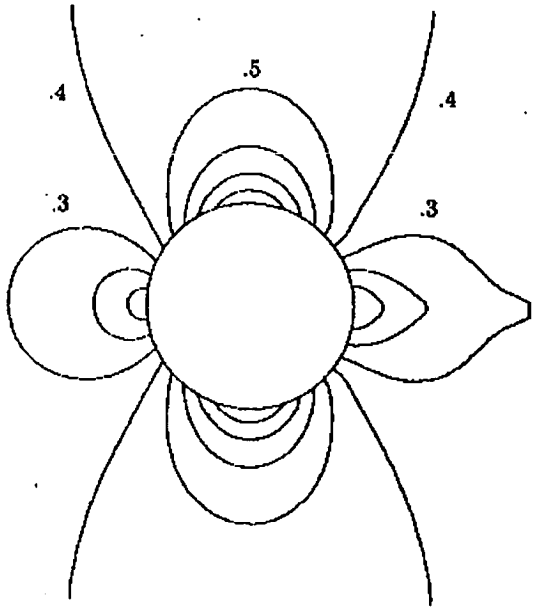


Figure 41: Compressible Cylinder Mach Contours No Curvature from [44]

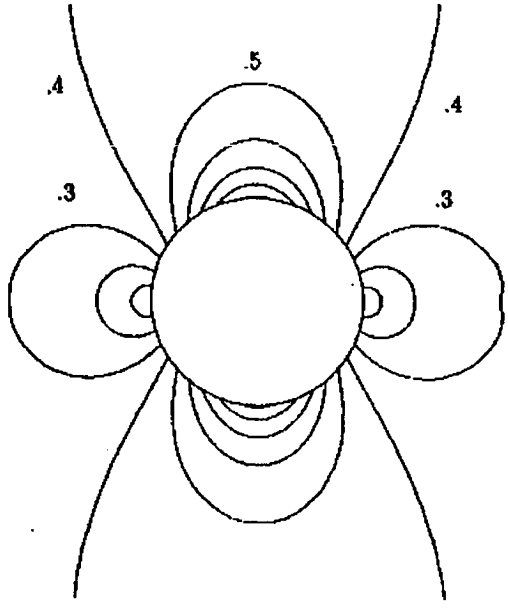


Figure 42: Compressible Cylinder Mach Contours with Curvature from [44]

Incompressible Viscous Flat Plate Flow

The incompressible flat plate case is a standard test case where the results can be compared to a known Blasius analytical solution, see [186] for details of the derivation. The flat plate is one unit long and oriented along the x-axis in a $M_\infty = 0.2$ freestream flow and a freestream Reynolds number of $Re_\infty = 10,000$. The computational boundaries extend 0.25 units in front of the leading edge and behind the trailing edge and 0.25 units above the flat plate. The solution is presented on a computational domain with a root grid dimension of 60x10 and 6 levels of refinement. In addition, the finest level of cells are within 0.008 units of the flat plate. The solution converged in approximately 40,000 iterations. The final grid consists of 52,926 cells. Figure 43 shows the final grid. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure.

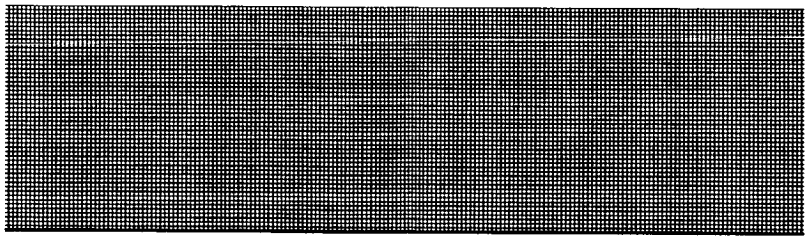


Figure 43: Final Computational Domain for Incompressible Flat Plate Flow

Figure 44 shows the skin friction coefficient for this case compared against the Blasius solution and the results from Coirier [38]. Generally, there is excellent agreement between the Blasius solution and the NASCART-GT solution. There are some differences at the leading edge of the flat plate that are caused by inadequate cell resolution for the very

small boundary layer region associated with the leading edge region. There is also a slight acceleration at the trailing edge due to the fact that the plate is not infinite that causes the skin friction coefficient to rise. Figure 45 shows the u-velocity profile through the boundary layer at the quarter-point and mid-point of the flat plate. Here excellent agreement is shown between the Blasius solution and the computed solution, and the computed solution shows the self-similarity that is expected.

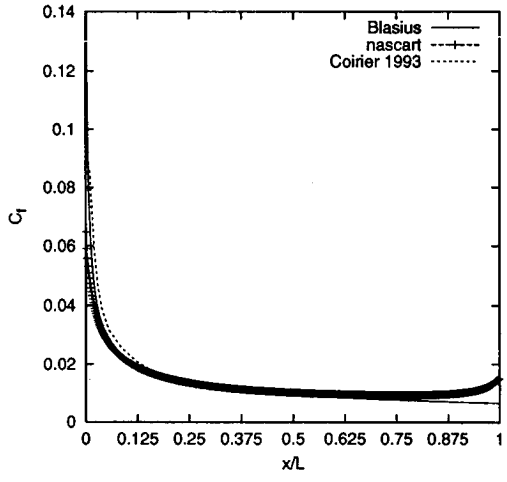


Figure 44: Incompressible Flat Plate Skin Friction Coefficient with Interpolated Reference Points

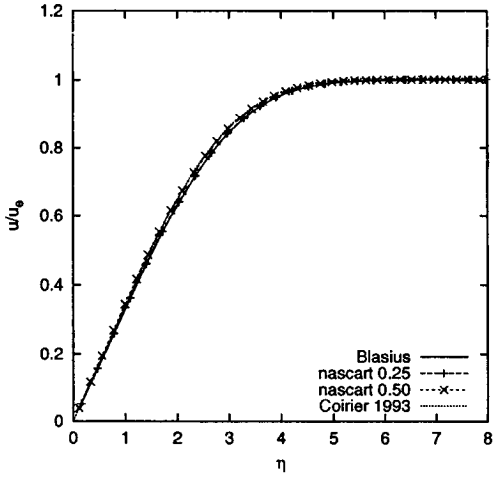


Figure 45: Incompressible Flat Plate Velocity Profiles with Interpolated Reference Points

Non-Grid Aligned Incompressible Viscous Flat Plate Flow

The non-grid aligned incompressible flat plate case is the same flow conditions as the incompressible flat plate flow case above. The difference is that for this case the flat plate and freestream velocity vector are at a 30° angle to the x-axis. The solution is still a Blasius solution, however now there are cut cells along the surface. The grid used for this solution is a 18x12 root grid dimension with 6 levels of refinement. The solution converged in

approximately 20,000 iterations. The final grid consists of approximately 13,152 cells. Figure 46 shows the final grid. For this case the reference points for the wall boundary conditions are determined without using the interpolation procedure.

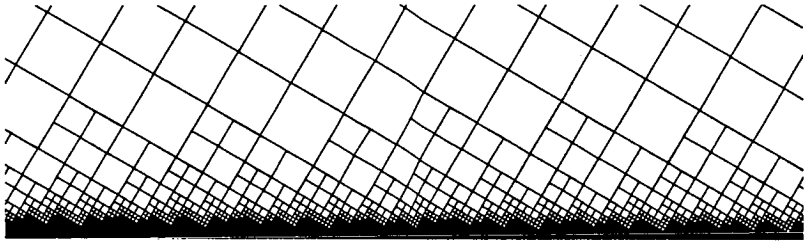


Figure 46: Final Computational Domain for Incompressible Non-Grid Aligned Flat Plate Flow

Figure 47 shows the skin friction coefficient for this case. Once the leading edge grid resolution problem is passed, at $x/L \approx 0.5$, there is good agreement between NASCART-GT and the Blasius solution. However, at the leading edge, the Blasius solution is not reliable due to the low local Reynolds number there, and the computed solution requires more grid points to adequately resolve this region. As in the above case, acceleration at the trailing edge caused by the finite length of the flat plate causes an increase in the skin friction coefficient.

The skin friction coefficient for this solution should follow the following curve

$$C_f = ax^b \tag{68}$$

where for the actual Blasius solution of this flow, the values for a and b are 0.00664 and -0.500, respectively. Using a standard nonlinear least-squares algorithm to minimize the errors between equation (68) and the NASCART-GT data results in the values of a and b of

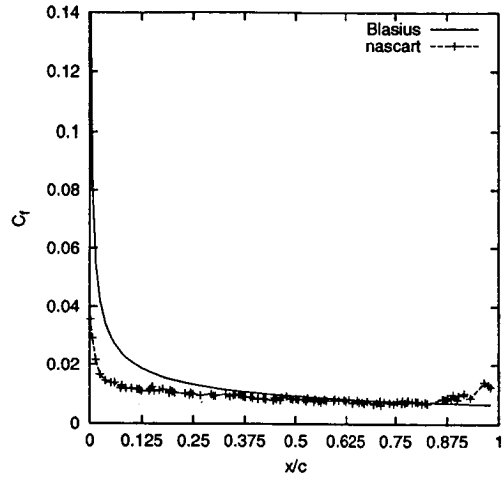


Figure 47: Incompressible Flat Plate Skin Friction Coefficient on Non-Grid Aligned Flat Plate without Interpolated Reference Points

0.00629 and -0.427 for the NASCART-GT results, respectively. Thus, in the region where the leading edge resolution and the trailing edge acceleration are not adversely effecting flow, the NASCART-GT solution maps quite closely to the Blasius solution.

Supersonic Viscous Flat Plate Flow

The supersonic flat plate case is another standard test case that has been extensively studied. The flat plate is one unit long and oriented along the x-axis in a $M_\infty = 3.0$ freestream flow and a freestream Reynolds number of $Re_\infty = 1000$. The computational boundaries extend 0.2 units in front of the leading edge and 0.8 units behind the trailing edge and 0.8 units above the flat plate. The solution is presented on a computational domain with a root grid dimension of 20x16 and 6 levels of refinement. In addition, solution adaption is performed every 1000 iterations. The solution converged in approximately 40,000 iterations with the CFL number at 0.10. The final grid consists of 15,337 cells. Figure 48 shows the final grid. For this case the reference points for the wall boundary conditions are determined without using the interpolation procedure.

The results from this case are compared with Arminjon and Madrane [11], Satya Sai et al. as well as the standard reference for the computational solution for an infinitely long flat plate, Carter [29]. The Satya Sai et al. results are for an infinitely long flat plate and are validated against Carter, and the Arminjon and Madrane results are for a finite length flat plate and are validated against Satya Sai et al.

Figure 49 shows the skin friction coefficient for this case. There is excellent agreement between the Carter results and NASCART-GT solution until the effects of the finite flat plate are seen around $x/L = 0.5$ in the NASCART-GT solution. The fact that the plate is finite causes the flow to accelerate as it approaches the trailing edge, thus the skin friction coefficient increases. Figure 50 shows the surface pressure for this case. Again there is

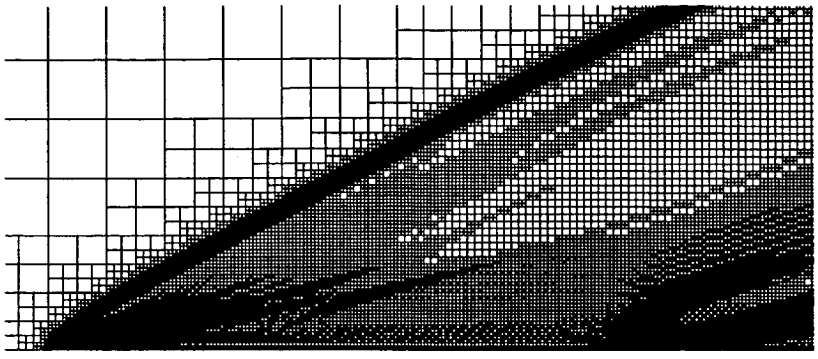


Figure 48: Final Computational Domain for Supersonic Flat Plate Flow

excellent agreement between the Satya Sai et al. results and the NASCART-GT solution until the trailing edge acceleration effects dominate. Notice that these effects appear further down the flat plate, $x/L = 0.75$, since the boundary layer pressure is less sensitive to the acceleration effects. Figure 51 shows the Mach contours for this case, and figure 52 shows

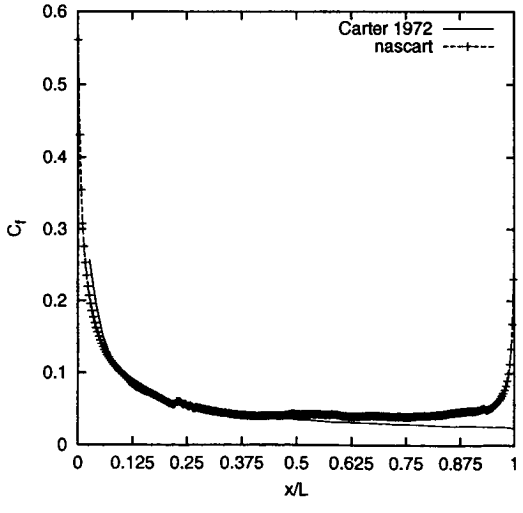


Figure 49: Supersonic Flat Plate Skin Friction Coefficient without Interpolated Reference Points

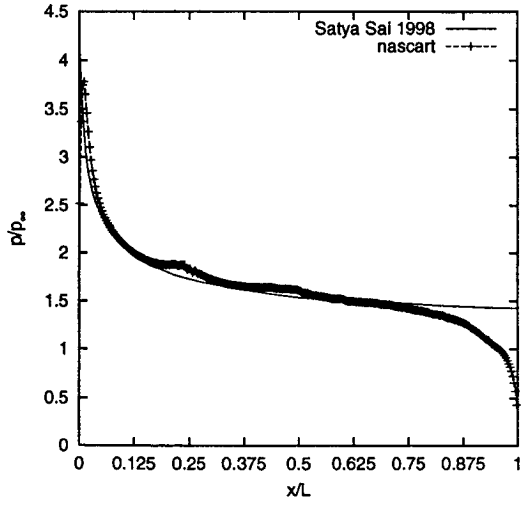


Figure 50: Supersonic Flat Plate Pressure without Interpolated Reference Points

the reference Mach contours from Arminjon and Madrane. There is excellent agreement

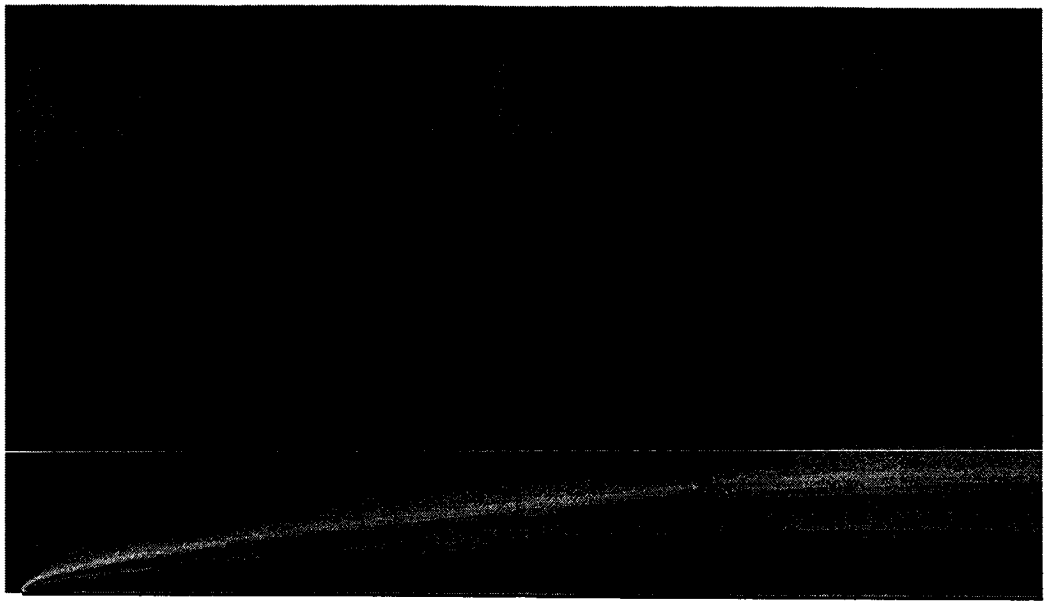


Figure 51: Mach Contours for Supersonic Flat Plate without Interpolated Reference Points

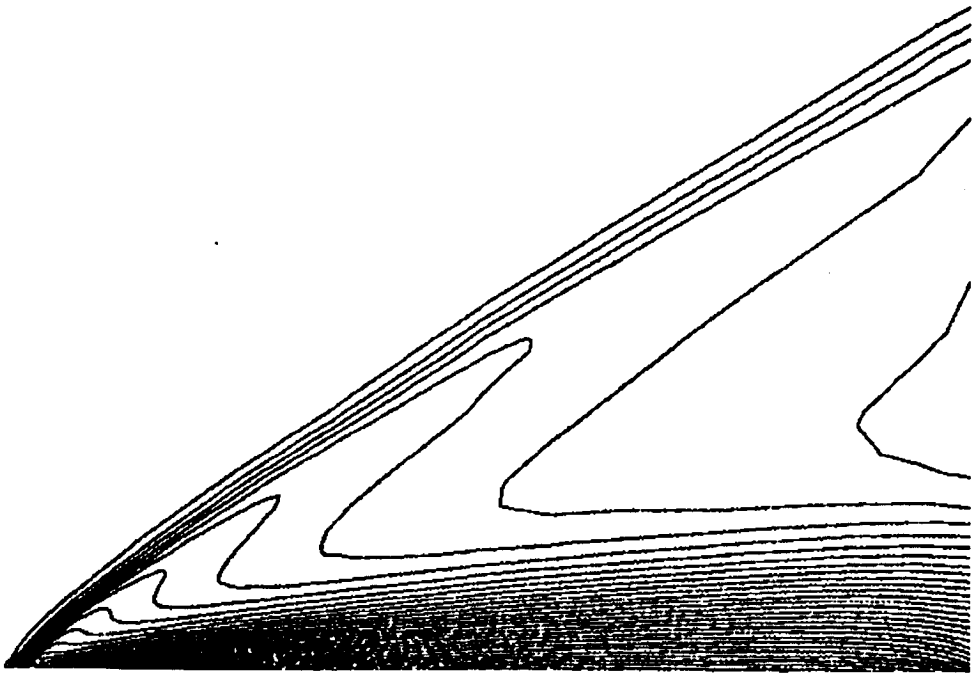


Figure 52: Supersonic Flat Plate Mach Contours from [11]

between the two contour plots with NASCART-GT crisply capturing the boundary layer induced shock as well as the boundary layer growth.

Two-Dimensional Airfoil Flows

The next set of cases are two-dimensional airfoil flows that have well studied computational solutions which can be used to further validate the code. The inviscid wall boundary conditions are compared to a transonic NACA-0012 airfoil flow, while the viscous wall boundary conditions are compared to a subsonic and supersonic NACA-0012 airfoil flow.

Transonic Inviscid NACA-0012 Airfoil Flow

This test case is a NACA-0012 airfoil in a $M_\infty = 0.85$ flow at an angle-of-attack of $\alpha = 1.00°$. The computational boundaries are 5 chords ahead and behind the airfoil and 5 chords above and below the airfoil centerline. Solutions are presented on a computational domain with a root grid dimension of 44x42 and 7 levels of refinement. In addition, solution adaption is performed every 200 iterations starting after 1000 iterations. Both solutions converged in approximately 20,000 iterations using local time-stepping. The final grids for the flat wall solution consists of 7981 cells and 7963 cells for the curved wall solution. Also, a curvature maximum of 40.0 is imposed in order to limit the large pressure gradients that can result near the leading edge. Figure 53 shows the final grid for the curved wall solution. Notice that the solution adaption has refined cells near the leading edge where the flow is going through rapid accelerations and near the shocks. The results from this

case are compared with the AGARD Advisory Report results [119] which presents general results from several researchers as well as detailed results for a 320x64 (20,480) cell structured grid solution. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure.

To further validate the curvature calculations of NASCART-GT, the exact curvature for the NACA-0012 airfoil, see [85] and Appendix C for details, is compared to the values obtained from NASCART-GT. Figure 54 shows the plot of the curvature and generally excellent agreement can be seen. There are slight differences between the the actual and computed curvatures at the leading edge, but these are due to using the minimum curvature for each computational cell that has multiple geometric intersections.

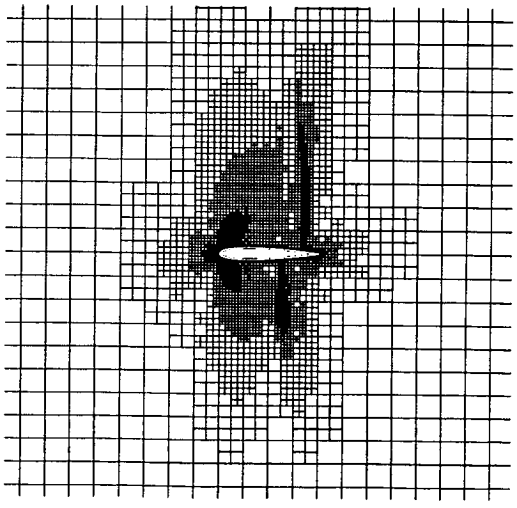


Figure 53: Final Computational Domain for Transonic Inviscid NACA-0012 Flow

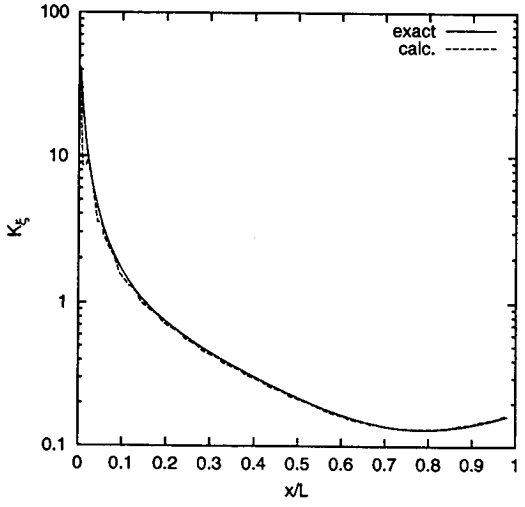


Figure 54: NACA-0012 Curvature Calculated from NASCART-GT

Figures 55 and 56 show the surface pressure coefficient comparison between the NASCART-GT solutions and the AGARD solution for the upper and lower surfaces, respectively. The curved wall solution does a better job of capturing the rapid accelerations with only slight

differences at the leading edge. The upper surface shock locations are missed by approximately 0.023 chords fore and 0.014 chords aft for the curved wall and flat wall solutions respectively. For the lower surface the curved wall solution is very close to the reference data, while the flat wall solution is approximately 0.028 chords aft.

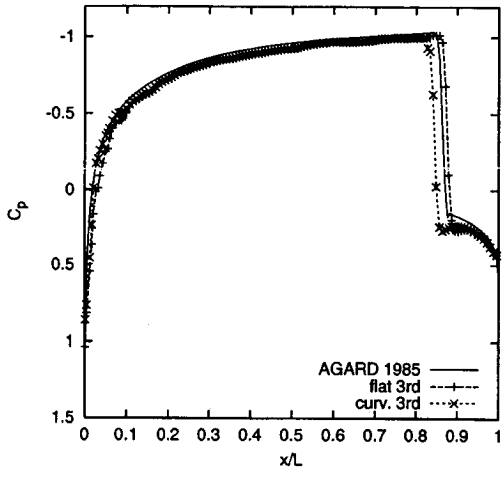


Figure 55: Transonic Inviscid NACA-0012 Upper Surface Pressure Coefficient with Interpolated Reference Points

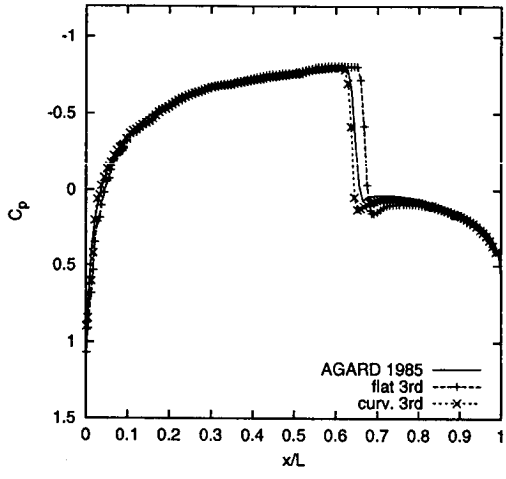


Figure 56: Transonic Inviscid NACA-0012 Lower Surface Pressure Coefficient with Interpolated Reference Points

Figures 57 and 58 show the Mach contours for the flat wall and curved wall solutions, respectively. Figure 59 shows the Mach contours from the AGARD reference. All three figures use a $\Delta M = 0.05$ for the contours. Both wall boundary conditions do an excellent job of capturing the flow features throughout the computational domain.

Finally, table 10 shows the lift and drag coefficients for the flat wall and curved wall cases as well as the AGARD committee results. In addition, the scatter associated with the various computed results by the AGARD researchers is also provided. The flat wall

boundary condition solution performs slightly better than the curved wall boundary condition solution for the lift coefficient with a 6.8% under-prediction versus 10.7% for the curved wall solution, however each result is within the scatter of the AGARD data. The curved wall boundary condition does a much better job at predicting the drag coefficient and is under the AGARD data by 7.4%. However, the flat wall boundary condition over-predicts the drag by 23%, but is close to the AGARD range. This is due to the inability of the flat wall to capture the leading edge suction peaks. Given the fact that NASCART-GT used only approximately 40% of the cells that the AGARD reference used, the curved wall results are quite reasonable.

Table 10: Transonic Inviscid NACA-0012 Lift and Drag Results

	flat wall	curved wall	AGARD [119] (scatter)
C_l	0.3341	0.3201	0.3584 (0.0589)
C_d	0.07150	0.05371	0.0580 (0.0126)

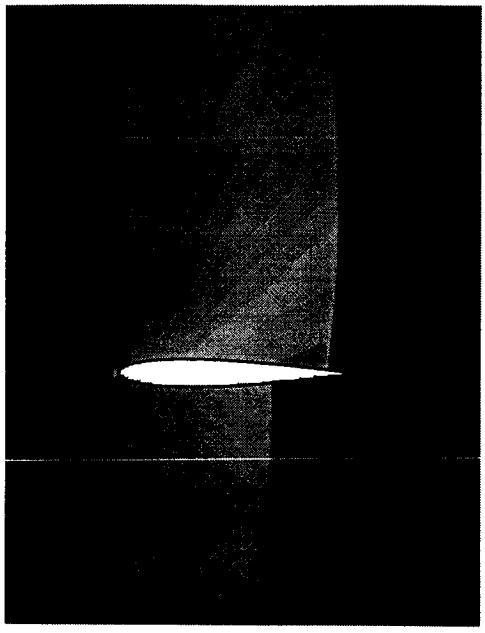


Figure 57: Mach Contours for Transonic Inviscid NACA-0012 Flat Wall

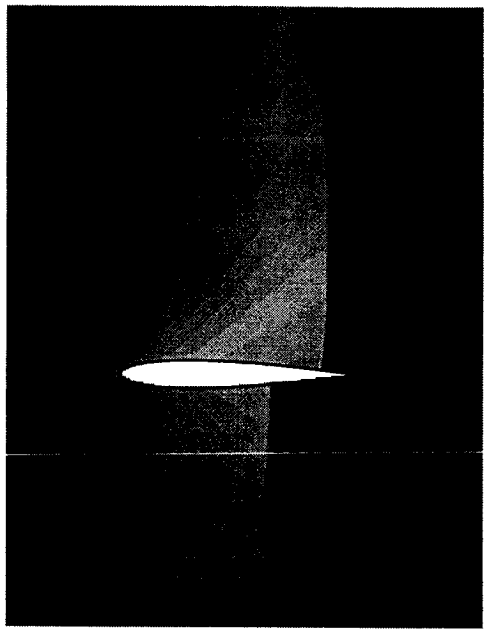


Figure 58: Mach Contours for Transonic Inviscid NACA-0012 Flow Curved Wall

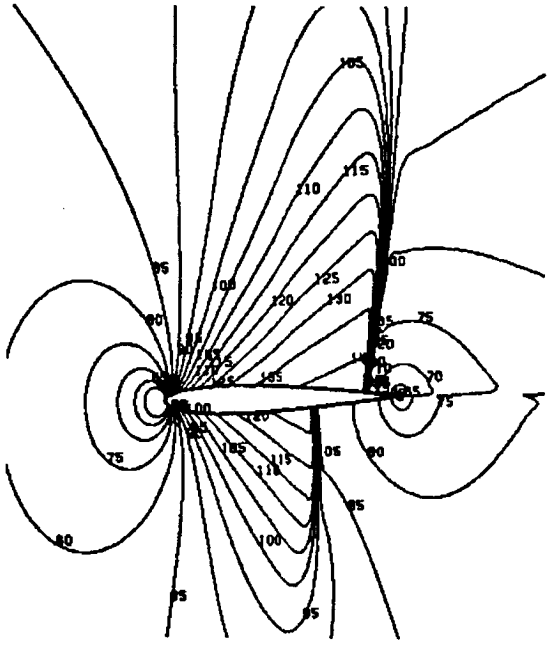


Figure 59: Inviscid Transonic NACA-0012 Mach Contours from [119]

Subsonic Viscous NACA-0012 Airfoil Flow

This test case is a NACA-0012 airfoil in a $M_\infty = 0.8$ flow at an angle-of-attack of $\alpha = 10°$ and a freestream Reynolds number of $Re_\infty = 500$. The computational boundaries are 5 chords ahead of the airfoil, behind the airfoil, above the airfoil centerline and below the airfoil centerline. Solutions are presented on a computational domain with a root grid dimension of 33x30 and 6 levels of refinement. In addition, solution adaption is performed every 500 iterations starting after 1000 iterations. Both solutions converged in approximately 40,000 iterations. The final grids for the flat wall solution consists of 57,100 cells and 56,947 cells for the curved wall solution. Also, a curvature maximum of 40.0 is imposed. Figure 60 shows the final grid for the curved wall solution. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure.

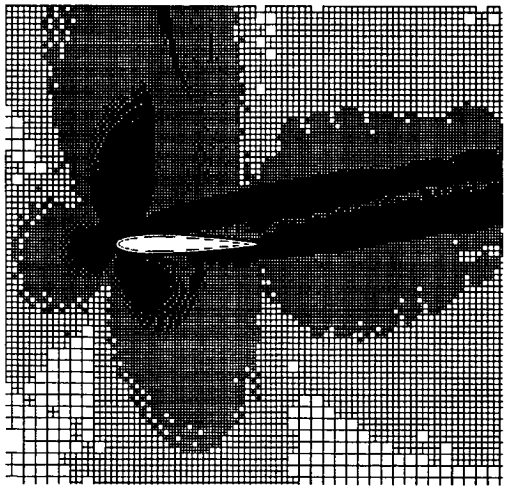


Figure 60: Final Computational Domain for Subsonic Viscous NACA-0012 Flow

The results from this case are compared with the results from Casalini and Dadone [32], whose results compare quite well to a collection of results from Bristeau et al. [27] others from an international workshop on compressible Navier-Stokes solvers. The Casalini and Dadone results are from a structured grid solution with 256x64 (16,384) cells.

Figure 61 shows the surface pressure coefficient comparison between the NASCART-GT solutions and the results from Casalini and Dadone. The flat wall and curved wall solutions show little differences between each other. They both capture the suction peak near the leading edge reasonably well, and slightly over-predict the lower surface pressure. In general, the agreement between the reference solution and the NASCART-GT surface pressure coefficient distributions is good.

Figure 62 shows the skin friction coefficient comparison between the NASCART-GT solutions and the results from Casalini and Dadone. Here, the leading edge skin friction coefficient is not well resolved until x/L of 0.1 on the upper surface and 0.15 on the lower surface. This is simply a grid resolution problem that would require multiple levels of grid cells along the body to reasonably capture the leading edge effects, which is currently not an option in NASCART-GT. Adding this functionality would require careful examination of the viscous stencil positivity criteria discussed by Coirier [38] in order to ensure that non-smoothness is not introduced into the solution. Notice that there are no large oscillations in the skin friction coefficient as was shown by other cut cell Cartesian approaches. Generally, after the leading edge resolution problem, there is excellent agreement between the reference skin friction coefficient and the NASCART-GT solutions.

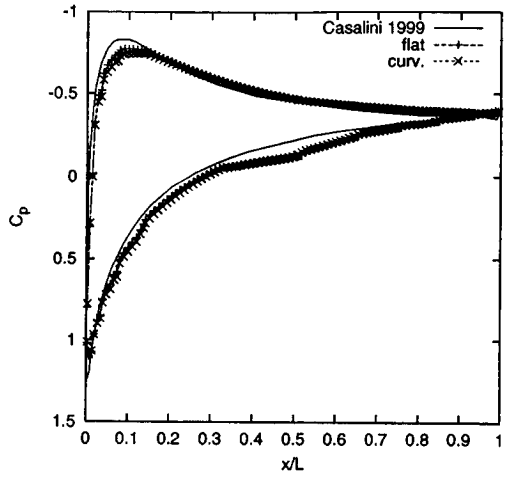


Figure 61: Subsonic Viscous NACA-0012 Surface Pressure Coefficient with Interpolated Reference Points

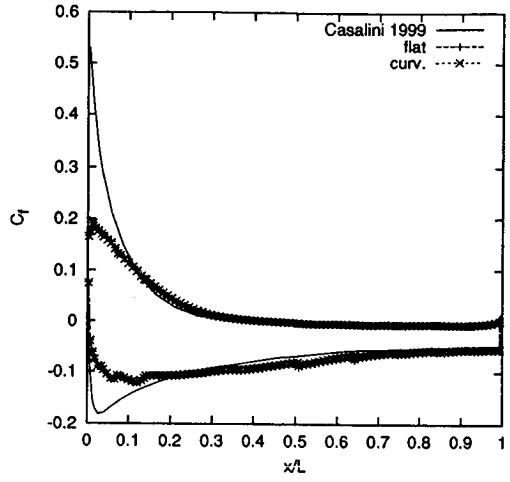


Figure 62: Subsonic Viscous NACA-0012 Skin Friction Coefficient Interpolated Reference Points

Figures 63 and 64 show the Mach contours for the flat wall and curved wall solutions, respectively. Figure 65 shows the Mach contours from the Casalini and Dadone reference. All three figures use a $\Delta M = 0.05$ for the contours. Both wall boundary conditions do an excellent job of capturing the flow features throughout the computational domain. In particular the recirculation region is clearly evident in both solutions. An examination of the skin friction coefficients for both solutions shows that the separation point occurs around x/L of 0.41 for the flat wall solution, which is 0.08 chords off of the location from Casalini and Dadone of 0.33, and 0.42 for the curved wall solution, which is 0.09 chords off.

Finally, table 11 shows the lift and drag coefficients for the flat wall and curved wall cases. These results are again compared to the Casalini and Dadone references mentioned above. The flat wall boundary condition over predicts the lift coefficient by 7.4% and

slightly under predicts the drag coefficient by 0.4%. The curved wall boundary condition also over predicts the the lift coefficient by 6.9% and slightly over predicts the drag coefficient by 0.4%.

Table 11: Subsonic Viscous NACA-0012 Lift and Drag Results

	flat wall	curved wall	Casalini and Dadone [32]
C_l	0.422	0.420	0.393
C_d	0.252	0.254	0.253

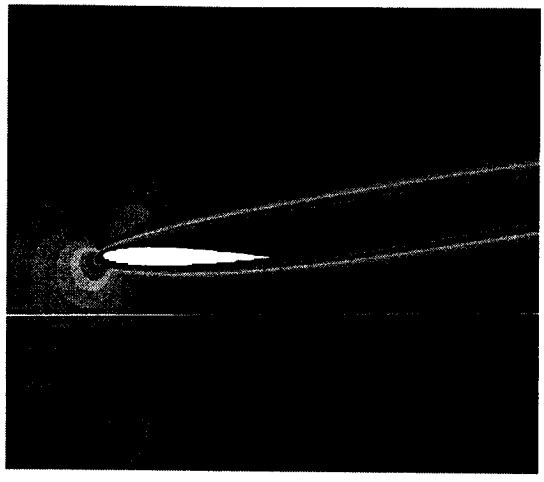


Figure 63: Mach Contours for Subsonic Viscous NACA-0012 Flow Flat Wall with Interpolated Reference Points

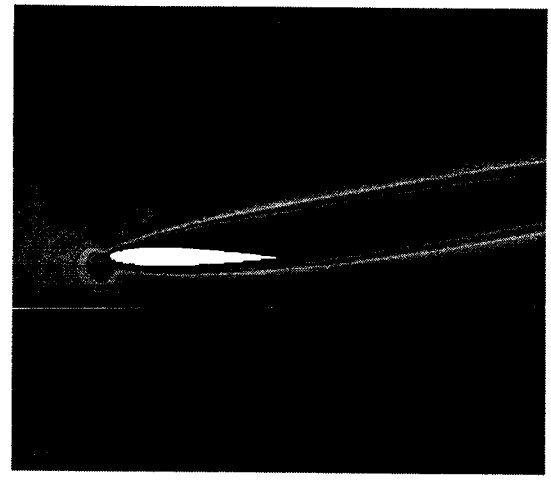


Figure 64: Mach Contours for Subsonic Viscous NACA-0012 Flow Curved Wall with Interpolated Reference Points

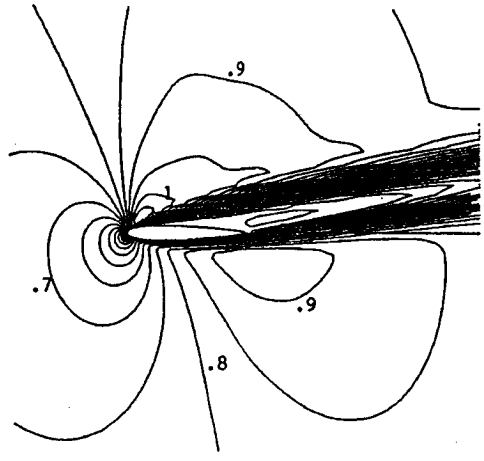


Figure 65: Viscous Subsonic NACA-0012 Mach Contours from [32]

Supersonic Viscous NACA-0012 Airfoil Flow

This test case is a NACA-0012 airfoil in a $M_\infty = 2.0$ flow at an angle-of-attack of $\alpha = 10°$ and a freestream Reynolds number of $Re_\infty = 1000$. The computational boundaries are 1 chord ahead of the airfoil, 6 chords behind the airfoil and 5 chords above and 3 chords below the airfoil centerline. Solutions are presented on a computational domain with a root grid dimension of 24x24 and 6 levels of refinement. In addition, solution adaption is performed every 200 iterations starting after 1000 iterations. Both solutions converged in approximately 20,000 iterations. The final grids for the flat wall solution consists of 47,741 cells and 48,088 cells for the curved wall solution. Also, a curvature maximum of 40.0 is imposed. Figure 66 shows the final grid for the curved wall solution. For this case the reference points for the wall boundary conditions are determined using the interpolation procedure.

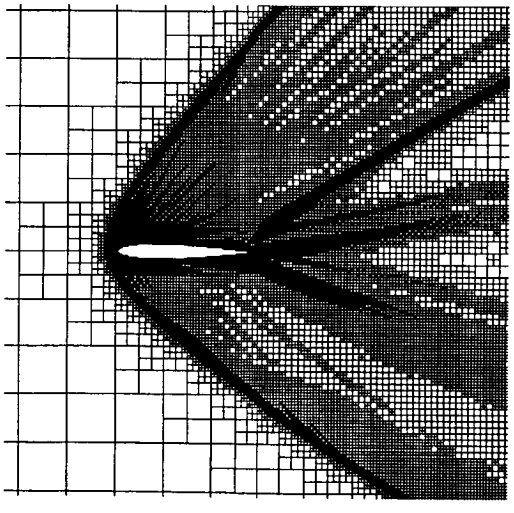


Figure 66: Final Computational Domain for Supersonic Viscous NACA-0012 Flow

The results from this case are compared with the results from Arminjon and Madrane [11], whose results compare quite well to a collection of results from Cambier [28] and Müller et al. [113] from an international workshop on compressible Navier-Stokes solvers. The Arminjon and Madrane results are from an unstructured grid solution with 7962 vertices. The Cambier results are from a structured grid solution with 193x72 (13,896) cells, and the Müller results are from a structured grid solution with 257x257 (66,049) cells.

Figure 67 shows the surface pressure coefficient comparison between the NASCART-GT solutions and the results from Arminjon and Madrane. Both solutions generally show excellent agreement with the reference data with slight differences on the upper and lower surfaces after about 0.1 chords for about 0.1 chords. In general, there is nice agreement between both solutions and the reference data and no significant differences between the curved wall or flat wall solutions.

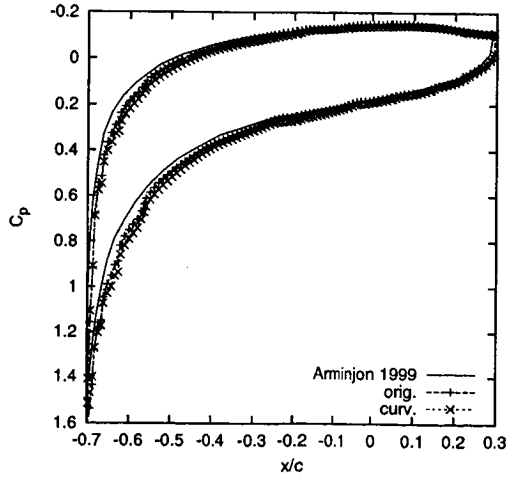


Figure 67: Supersonic Viscous NACA-0012 Surface Pressure Coefficient

Figures 68 and 69 show the Mach contours for the flat wall and curved wall solutions, respectively. Figure 70 shows the Mach contours from the Arminjon and Madrane reference. All three figures use a $\Delta M = 0.1$ for the contours. Both wall boundary condition cases do an excellent job of capturing the flow features throughout the computational domain. The bow shock is crisply captured in both solutions without any noticeable oscillations.

Finally, table 12 shows the lift and drag coefficients for the flat wall and curved wall cases. These results are compared to the Cambier and Müller references mentioned above. The flat wall boundary condition slightly under-predicts the lift coefficient by 1.8% compared to Cambier and by 0.7% compared to Müller. For the drag coefficient, the flat wall over-predicts both results, by 1.8% and 2.7%. The curved wall boundary condition is between the results of Cambier and Müller with a relative difference of 0.4% and 0.8% respectively. For the drag coefficient, the curved wall boundary condition slightly over-predicted by 0.7% and 1.7%.

Table 12: Supersonic Viscous NACA-0012 Lift and Drag Results

	flat wall	curved wall	Cambier [28]	Müller [113]
C_l	0.3364	0.3415	0.3427	0.3388
C_d	0.2583	0.2554	0.2535	0.2515

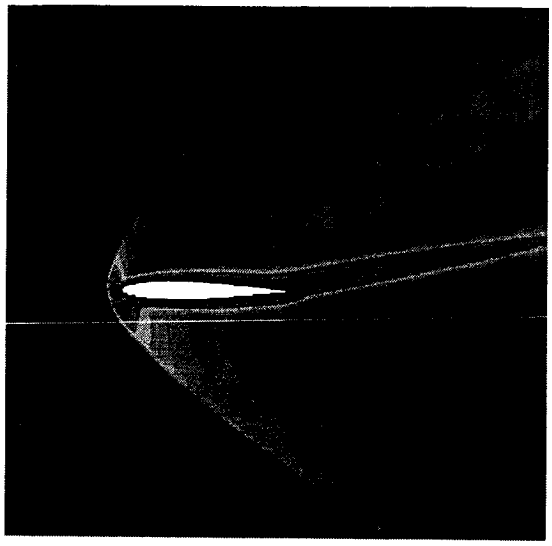


Figure 68: Mach Contours for Supersonic Viscous NACA-0012 Flat Wall with Interpolated Reference Points

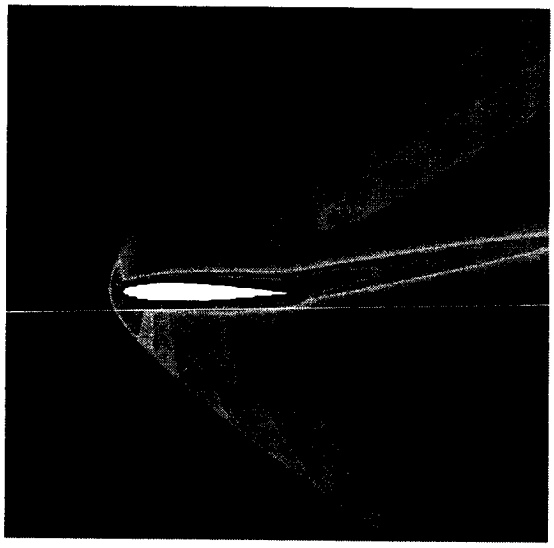


Figure 69: Mach Contours for Supersonic Viscous NACA-0012 Flow Curved Wall with Interpolated Reference Points

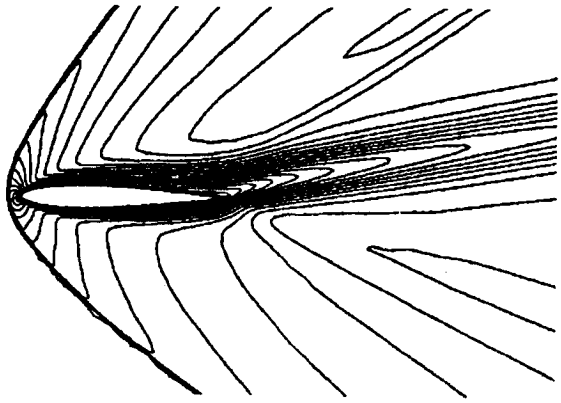


Figure 70: Viscous Supersonic NACA-0012 Mach Contours from [11]

Transonic Inviscid ONERA M6 Wing

This test case is an inviscid flow around an ONERA M6 wing in a $M_\infty = 0.84$ flow at an angle-of-attack of $\alpha = 3.06°$. The computational boundaries are 4 root chord lengths away in the x-, y- and z-directions. The solution is presented on a computational domain with a root grid dimension of 34x34x34 and 6 levels of refinement. In addition, solution adaption is performed every 500 iterations starting after 1000 iterations. The solution presented is after approximately 5300 iterations. The final grid for this case consists of 404,400 cells with 21,556 surface cells. As in the NACA-0012 cases, a curvature maximum of 40.0 is imposed in order to limit the pressure gradients caused by the highly curved regions of the leading edge. Figure 71 shows the final grid for this case. For this case the reference points for the wall boundary conditions are determined without using the interpolation procedure.

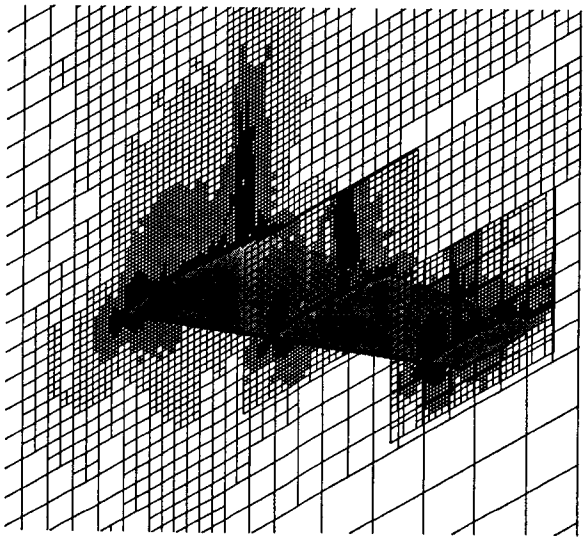


Figure 71: Final Computational Domain for Transonic Inviscid ONERA M6 Flow

The results from this case are compared with the results from AGARD Advisory Report (AGARD-AR-138) results [146] and AGARD Advisory Report (AGARD-AR-211) results [119]. The AGARD-AR-138 data is experimental data performed for a very high Reynolds number, 11.72×10^6, in order to minimize the displacement thickness effects caused by the boundary layer. The AGARD-AR-211 data is a collection of computational results from several researchers for an inviscid solution to this problem. The AGARD-AR-211 computational results have significantly more resolution at the leading edge compared to the NASCART-GT geometry with approximately 4 cells from the AGARD fine grid solution fitting into the leading edge cell of the NASCART-GT geometry. However, once the leading edge section is passed, the cell sizes between the fine AGARD computational results and the NASCART-GT geometry are nearly equal. Thus, it is reasonable to expect that the leading edge resolution of the NASCART-GT results will not be as accurate as the AGARD computational results.

Figures 72 through 77 show the surface pressure values at several span-wise locations for the NASCART-GT solution and the AGARD-AR-138 results. As with many of the other cases presented above, more leading edge resolution is needed in order to accurately capture the rapid suction peaks, especially near the root of the wing on the upper surface. As is typical in inviscid solutions [3], the upper surface shock locations are slightly aft of the experimental results due to the neglect of the boundary layer effects. For the inboard sections, figures 72 through 75, there are two separate shocks on the upper surface that are present in the experimental results, however the inadequate leading edge resolution

prevents the capturing of the first. After the first shock, there is better agreement. The lower surface shows excellent agreement throughout all of the figures.

A direction comparison of the NASCART-GT results with other inviscid solutions is difficult because other solution techniques are not limited to a single cell size throughout the entire solid surface as is NASCART-GT in order to property handle the modeling of viscous flows. However, other inviscid solutions also predict the stronger shock location aft of the experimental location, for example [3] as well as the AGARD-AR-211 computational results, with generally good agreement with the NASCART-GT locations.

Figures 78 and 79 show the Mach contours on the upper surface of the wing for NASCART-GT and the AGARD-AR-211 results, respectively. Both figures use a $\Delta M = 0.05$ for the contours. In these figures it is apparent that there is a lambda-shock structure on the upper surface with NASCART-GT only capturing the second shock and the top of the lambda. It appears that the first shock, the weaker of the two, is close to forming in the NASCART-GT solution.

Figures 80 and 81 show the Mach contours on the lower surface of the wing for NASCART-GT and the AGARD-AR-211 results, respectively. Both figures use a $\Delta M = 0.05$ for the contours. Here there is nice agreement between the two results with only slight differences in the center of the mid-span region where there is some discontinuity in the NASCART-GT contours.

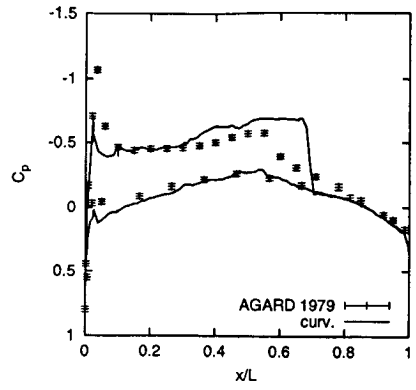


Figure 72: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.2$ without Interpolated Reference Points

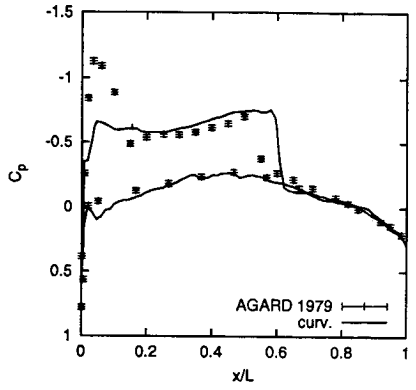


Figure 73: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.44$ without Interpolated Reference Points

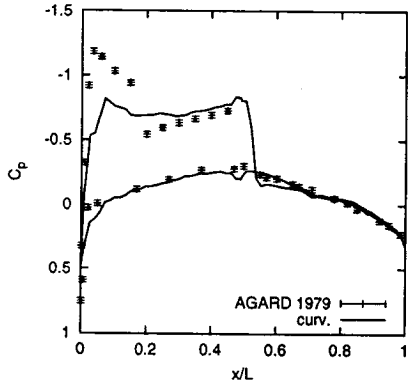


Figure 74: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.65$ without Interpolated Reference Points

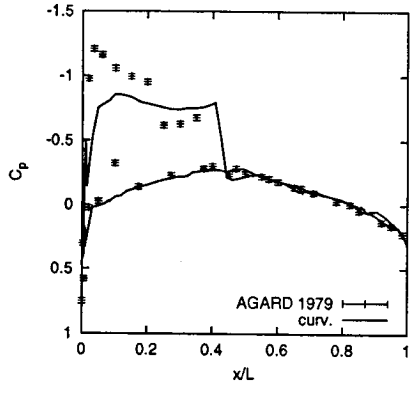


Figure 75: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.8$ without Interpolated Reference Points

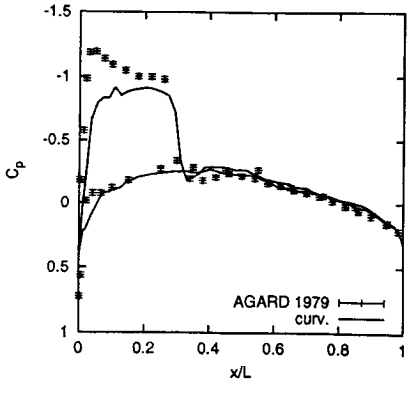


Figure 76: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.9$ without Interpolated Reference Points

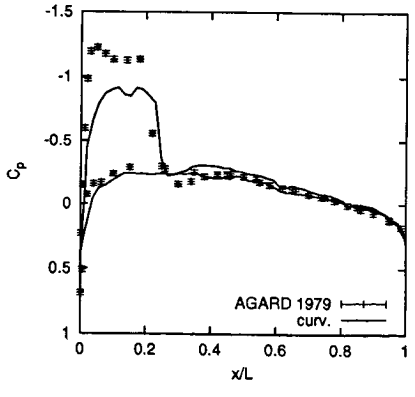


Figure 77: Transonic Inviscid ONERA M6 Surface Pressure Coefficient at $z/L = 0.95$ without Interpolated Reference Points

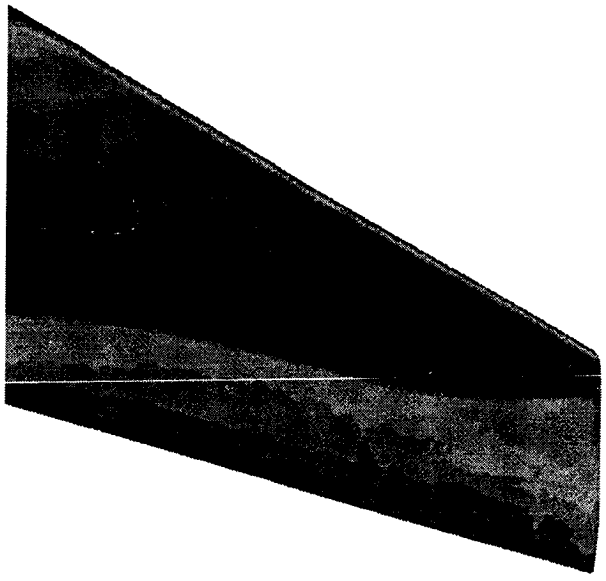


Figure 78: Transonic Inviscid ONERA M6 Upper Surface Mach Contours without Interpolated Reference Points

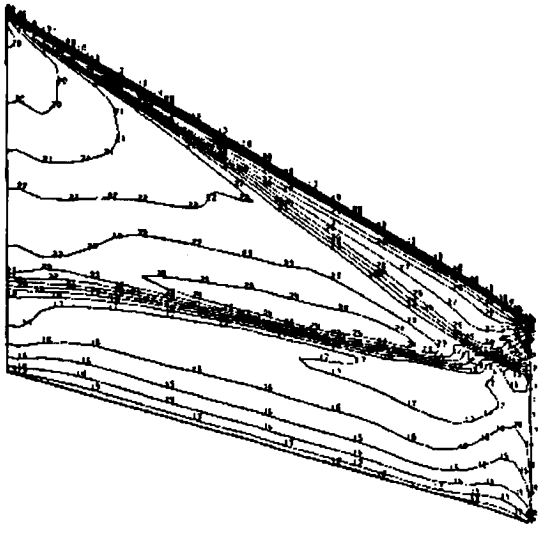


Figure 79: Transonic Inviscid ONERA M6 Upper Surface Mach Contours from [119]

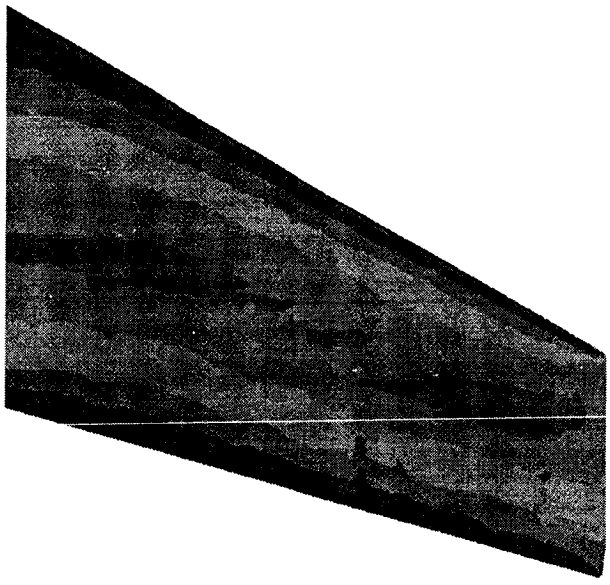


Figure 80: Transonic Inviscid ONERA M6 Lower Surface Mach Contours without Interpolated Reference Points

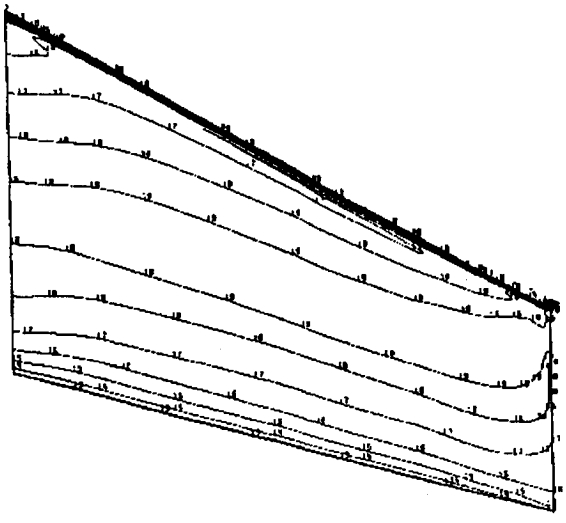


Figure 81: Transonic Inviscid ONERA M6 Lower Surface Mach Contours from [119]

CHAPTER VI

PARALLELIZATION RESULTS

With the modifications made to flowCart (the flow solver part of CART3D) mentioned in Chapter IV, tests were performed to demonstrate the parallelization characteristics of the MPI version of flowCart. This chapter discusses the parallelization performance of the MPI version of flowCart and compares the results to the OpenMP version as well as other published results for similar configurations.

Test Hardware Description

There were two separate hardware configurations used to test the MPI parallelization enhancements, the first was an Origin 2000 for the shared memory tests, and the second was a heterogeneous cluster of SGI workstations connected by Gigabit ethernet for the distributed memory tests.

Shared Memory System Configuration

The shared memory hardware used for these tests was part of NASA Ames Research Center's NAS (NASA Advanced Supercomputing) Division CoSMO/NAS/HPCCP clusters. The machine, Lomax [117, 118], was a 256 node Origin 2000 with 2 400 MHz R12000

CPUs per node for a total of 512 available processors. Each node contained 768 MB of memory (with approximately 700 MB available for application use) for a total of 192 GB of memory. Each node also contained 32 KB of on-chip L1 cache and 8 MB of external L2 cache. The memory hierarchy was as follows:

- CPU registers
- L1 instruction cache and data cache
- L2 unified (instruction and data) cache
- Local main memory
- Remote main memory
- Hard disk

with the latency associated with memory accesses increasing down the list.

The operating system on Lomax was SGI Irix v6.5.10f. The executables were compiled with SGI MIPSPro FORTRAN 77 and C compilers v7.3.1.1m using the `-Ofast` optimization flag in 64-bit mode. The OpenMP and MPI parallelization libraries used were the libraries supplied by SGI Message Passing Toolkit v1.4.0.0.

Distributed Memory System Configuration

The distributed memory hardware used for these test was a cluster of SGI workstations at NASA Ames Research Center. The cluster, Cluster T27B [116], was composed of 19

SGI workstations, 14 Octane and 5 Octane2 machines, with processor speeds varying from 250 MHz to 400 MHz and available memory between 896 MB to 3584 MB (see Table 13 for the configuration of the specific machines). The cluster was connected using gigabit ethernet.

Table 13: Distributed Memory Cluster Information

Machine	Processor Type	Processor Speed (MHz)	Memory (MB)	OS
Octane	1 × R10000	250	1280	IRIX v6.5.13m
Octane	2 × R10000	250	2048	IRIX v6.5.13m
Octane	2 × R10000	250	2048	IRIX v6.5.13m
Octane	1 × R12000	300	896	IRIX v6.5.13m
Octane	1 × R12000	300	1024	IRIX v6.5.13m
Octane	1 × R12000	300	2048	IRIX v6.5.13m
Octane	1 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane	2 × R12000	300	2048	IRIX v6.5.13m
Octane2	2 × R12000	360	2304	IRIX v6.5.13m
Octane2	2 × R12000	360	2304	IRIX v6.5.13m
Octane2	2 × R12000	360	2304	IRIX v6.5.13m
Octane2	2 × R12000	360	3584	IRIX v6.5.13m
Octane2	2 × R12000	400	2304	IRIX v6.5.14m

The operating system on each of the machines was SGI IRIX v6.5.13m (except for one Octane2 machine which had SGI IRIX v6.5.14m, see Figure 13). The executables were compiled with SGI MIPSPro FORTRAN 77 and C compilers v7.3.1.2m using the `-Ofast`

optimization flag in 64-bit mode. The MPI parallelization library used was the MPICH [65] library v1.2.1.

Parallelization Quantization Methodology

In order to provide an accurate assessment of the peak performance of flowCart in a parallel processing environment, the following procedures were used to create the results. In order to objectively compare the parallelization results, the same processors needed to be used for the entire range of speedup cases. Thus, the maximum number of processors to be used was allocated at the beginning of the tests and each speedup case used a subset of these processors. Since there was no guarantee that the optimal processor allocation would be obtained for any particular run, three runs of 20 iterations were performed for each set of processors and the best timing was taken. This also minimized the effects of any memory bandwidth and CPU contention caused by other users on the systems. Finally, to remove any one-time initialization costs, the reported time for each run was taken to be the time for the 1^{st} iteration subtracted from the 20^{th} iteration. The elapsed time for each iteration was recorded using the standard UNIX function `getrusage` to get the elapsed user time for the process with microsecond resolution.

Shared Memory Results

Since some of the modifications made to flowCart were to the core functionality (such as the overlap control volume exchange data structures discussed on page 97), a comparison

between the new OpenMP functionality and existing parallelization results was performed. Figure 83 shows the speedup for the new flowCart-OpenMP code using up to 64 processors compared to Berger et al. [22] results (labeled Berger-2000), Mavriplis [98] results (labeled Mavriplis-2000) and the ideal speedup (labeled Ideal). The flowCart-OpenMP and Berger-2000 cases used approximately 1.0 million control volumes, while the Mavriplis case used approximately 3.1 million control volumes. As can be seen in Figure 83, there is excellent agreement between all three cases with a slight decrease in performance for the 32 node case which is most likely cased by a poor distribution of the allocated nodes over the processors. Analyzing the run times for the 32 node flowCart-OpenMP result shows a wide variety between the slowest run (66.669 s) and the fastest run (51.463 s), which results in a 30% difference between the these two cases, while the other runs typically had a 7% difference between their slowest and fastest time.

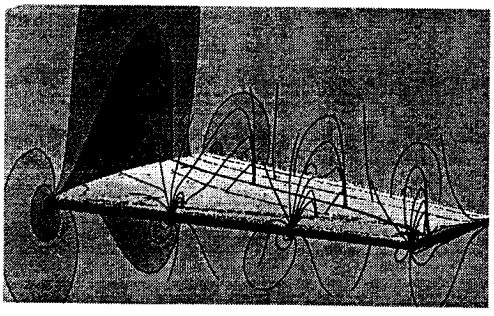


Figure 82: Sample Solution of ONERA M6 Wing Parallelization Case

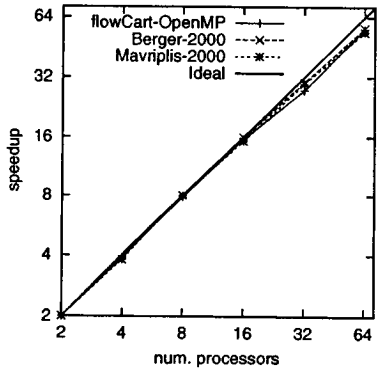


Figure 83: OpenMP Speedup Results Compared to Published Data

Figure 84 shows a comparison between flowCart-OpenMP and flowCart-MPI using the same 1.0 million control volume grid used above. For up to 16 processors, the speedup

curves are quite similar. For the 32 processor case, both sets of results begin to deteriorate due to the poor distribution of processors mentioned above, with flowCart-MPI showing less degradation in performance. For the 64 processor case, both speedup curves show improvements compared to the 32 processor case, with flowCart-MPI showing super-linear speedup. This is caused by the fact that the partition sizes are very small (approximately 16,000 control volumes/processor). Thus, most of the data can exist in the processor's cache, resulting in significantly less time required to access data than if the data resided in the nodes local memory. This super-linear speedup has also been demonstrated by other researchers [98] as shown in Figure 86. This effect is less pronounced for flowCart-OpenMP since it utilizes pointers for the IPC and not shared memory buffers as MPI. This also explains why flowCart-MPI does not show as drastic a penalty as flowCart-OpenMP does for the 32 processor case.

One final comparison of interest between flowCart-OpenMP and flowCart-MPI is the timing results, Figure 85. Overall flowCart-MPI is within 5% of the flowCart-OpenMP times except for the 64 processor case where the cache benefits discussed above result in flowCart-MPI being 15% quicker than flowCart-OpenMP, see Table 14. This result seems counter-intuitive since flowCart-MPI is at the very least having to perform a buffer fill and empty (assuming that the buffer exchange occurs as a shared-memory operation) while flowCart-OpenMP does not. The most likely cause is that as the number of control volumes per processor decreases, there is going to be many short requests for memory addresses in

the flowCart-OpenMP due to the way that information is exchanged, while the flowCart-MPI information exchange occurs as a few large blocks of data. Thus memory contention might become more of bottleneck for flowCart-OpenMP when a relatively large fraction of the control volumes are on processor boundaries.

Table 14 also demonstrates the improvements due to the cache benefits that have been observed in other figures.

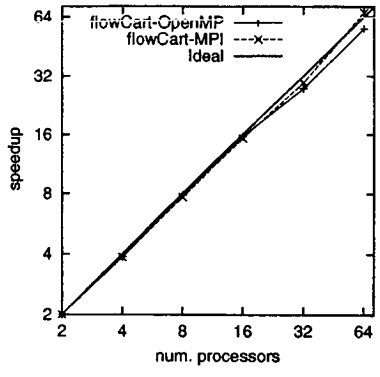


Figure 84: Shared Memory OpenMP and MPI Speedup Results

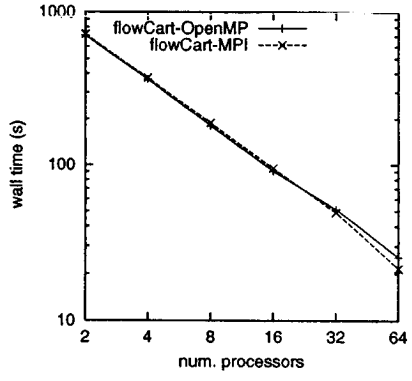


Figure 85: Shared Memory OpenMP and MPI Timing Results

Table 14: Shared Memory Timing Improvements for flowCart-MPI

num. proc.	% Improvement
2	-1.9
4	-2.1
8	+4.1
16	+4.1
32	+4.9
64	+15.0

Finally, Figure 86 shows a comparison between a 3.1 million control volume case from

Mavriplis [98] using MPI and a 1.0 million control volume case from flowCart-MPI. Again, there is good agreement between the two cases with the performance from Mavriplis showing slightly better speedup due to the larger grid and the additional computations that are being performed (viscous terms, GMRES, etc.). For the 64 processor case both curves show the same super-linear speedup caused by the cache benefits.

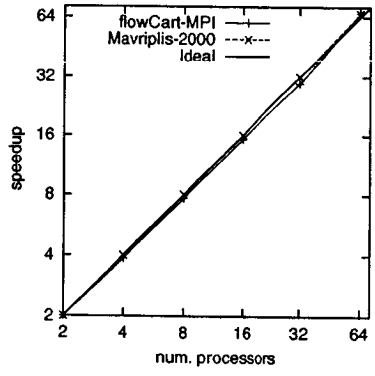


Figure 86: Shared Memory MPI Speedup Results Compared to Published Data

Distributed Memory Results

The distributed memory configuration results here are compared with the shared memory results obtained from flowCart-MPI for the same 1.0 million control discussed above. Figure 87 shows the speedup results. Acceptable parallelization performance is demonstrated up to 8 processors. After that point, the communication costs begin to overwhelm the computational benefits for 16 processors. Figure 88 shows that there is only a 15% performance penalty for using the distributed memory architecture until 8 processors. After

that, the communication costs again overwhelm the computations. Luecke et al. [94] as well as Kremenetsky et al. [83] have demonstrated that there is a significant performance penalty using the MPICH MPI library compared to using the SGI MPI library for both performance benchmarking applications as well as similarly sized CFD simulations. This seems to explain the relatively poorer distributed memory performance results compared to the shared memory results since the MPI version performs well in the SGI shared memory architecture.

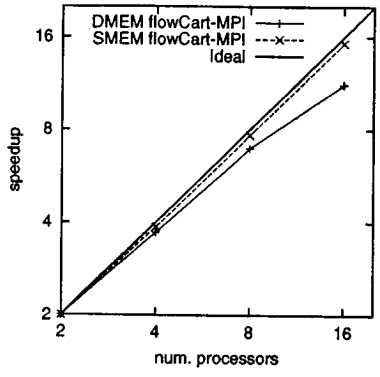


Figure 87: Distributed Memory MPI Speedup Results

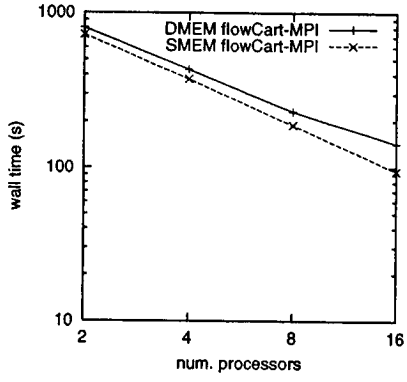


Figure 88: Distributed Memory MPI Timing Results

CHAPTER VII

CONCLUSIONS

This research has provided insight into ways of extending the functionalities of Cartesian grid solvers into viscous effects modeling via novel boundary condition treatments and MPI parallelization. The non-smoothness associated with the non-positivity of the viscous flux stencil for the surface cells have been minimized in NASCART-GT by separating the surface cells from the finite volume formulation that is used to solve the rest of the computational domain. While the surface cells are not part of the finite volume formulation, their state is still determined by applying physically based conditions that are consistent with the boundary conditions associated with the surface. Additionally, the parallelization functionality of CART3D has been extended to use MPI as its parallelization library without significant impact to the parallelization speedup or total run time.

Solid Boundary Treatment

The new viscous solid boundary treatment developed for NASCART-GT removes the surface cells from the finite volume formulation in order to address the non-smoothness and small time steps associated with the cut cell treatment. The state at the surface cells in determined by applying interpolation functions and the solid surface boundary conditions with

either flat or curved wall approximations. This new treatment shows significant progress towards utilizing cut cell Cartesian grid methods for general bodies in viscous flows. In all cases presented, the interpolation formulations produce reasonable results without the non-smoothness problems associated with the stencil positivity in the viscous cases. The integrated quantities of lift and drag are well predicted with both the flat wall and curved wall boundary conditions, with the curved wall boundary conditions typically producing slightly better results. The solid surface quantities compare well to existing results, with some cases showing difficulties near the leading edge. This difficulty is caused by the uniform surface cell size limitation imposed by the viscous scheme in order to avoid the viscous stencil positivity problem. Even when the leading edge region is not captured accurately, the curved wall boundary condition does a better job of predicting the surface features.

In terms of capturing the overall flow field characteristics, both schemes performed well in all cases. In general, the curved wall boundary condition formulations have improved the ability to capture the surface quantities in the highly curved regions of the surface for the inviscid cases and produced only marginal improvements in the viscous results. The fluctuations in the pressure and skin friction coefficients have been nearly eliminated by the use of the interpolated reference points in the boundary condition formulations.

These results indicate that the original algorithmic problem of solving the Navier-Stokes equations on Cartesian grids due to the viscous stencil positivity has been converted into a computational problem of being able to allocate enough memory and CPU time to

adequately resolve the entire surface. At the same time, the inviscid formulations on Cartesian grids can take advantage of the less stringent time step restrictions by removing the small cut cells from the finite volume formulation.

Parallelization Enhancements

The parallelization enhancements performed on CART3D demonstrate a conversion of a domain-decomposition flow solver implemented with OpenMP to a strict MPI message-passing structure. In all cases the MPI version performed as well as, or better than the already good performance of the OpenMP implementation. Moreover, the MPI parallelization performance also compares well to other published results. Near linear speedup has been demonstrated for up to 64 processors with a 1.0 million control volume grid using the MPI parallelization without adversely affecting the wall-clock timings for shared memory architectures, while reasonable speedups have been demonstrated for similar solutions on a distributed memory architecture. Using MPI for the parallelization library allows CART3D to be used in a shared memory environment without any performance penalties compared to OpenMP, as well as in a distributed memory environment where the OpenMP version was not able to be used.

Three-Dimensional Viscous Modeling

The final question in this research is how feasible is it to solve the Navier-Stokes equations for three-dimensional bodies using this new surface cell treatment. In order

to answer this question a quick analysis of the current performance of NASCART-GT is needed. Using similar techniques to check the timing of NASCART-GT that were used in the parallelization performance study of CART3D, the compute time for NASCART-GT is approximately 4.0×10^{-4} s/cell/iteration on a 500 MHz AMD-K6® processor (ignoring grid generation and file input/output times). This value scales linearly with the number of cells and the number of iterations. Assuming that the compute timings can be halved by upgrading to higher quality components (such as faster memory as well as faster and more up-to-date CPU) and a factor of five improvement from performance acceleration techniques (such as multigrid, GMRES and higher order temporal integration), then the amount of time needed for NASCART-GT to compute one cell in one iteration is approximately 4.0×10^{-5} s/cell/iteration. Assuming that a reasonable geometry can be modeled using 10 million cells (a conservative number in general, but certainly appropriate for low Reynolds number, $Re \approx 1000$, simple three-dimensional geometry flows) and that 50,000 iterations are required, then the amount of time it would take to solve the case is approximately 240 cpu-days. Now, taking the parallelization speedup results that a 1 million cell case can scale near linearly up to 64 processors and extrapolate that out to a 10 million cell case that has more computations per iteration, then it is reasonable to expect near linear speedups for 640 processors for this case (ignoring bandwidth limitations and other hardware related issues). Using these numbers, then an efficient, parallelized NASCART-GT solving a 10 million cell problem on a computational environment using current state-of-the-art hardware is projected to be possible in approximately 9 cpu-hours.

This is a reasonable turn-around time for full three-dimensional viscous flows. However, to model a complete flight vehicle at a reasonable Reynolds number, $Re \approx 10^7$, might require as much as 40 million cells or more. This means that a complete flight vehicle could take 2 cpu-days to complete, a less reasonable but still manageable amount of time. Thus, it is imperative that parallelization be utilized along side the new surface cell methodology in three-dimensional Navier-Stokes Cartesian solver along with aggressive acceleration techniques in order to solve a three-dimensional viscous flow.

Future Work

This research has shown that the two most common current limitations in Cartesian grid solvers have been addressed, however there are more improvements in both areas that can be accomplished in future work.

Extending the Current Surface Cell Modeling

While, these results show significant improvements in the handling of viscous solutions on Cartesian grids, there are several areas of research that need to be examined further. In order to address the accuracy problems in the leading edge regions of the surface, the functionality of having multiple levels of refinement on the surface needs to be added to NASCART-GT. This needs to be carefully studied since Coirier showed non-smoothness problems can arise even in regions where the cell sizes change is comparable to the changes at a refinement boundary. One possible approach to these surface refinement regions is to

use a viscous flux reconstruction stencil based on the modified diamond-path Green-Gauss developed by Delanaye et al. [49].

In an effort to improve the accuracy of the interpolation formulations, more sophisticated wall modeling techniques should be investigated. Specifically, modeling the states along the interpolation line with analytical solutions, such as analytical boundary layer modeling, should be studied. In addition, extending the applicable range of solutions from laminar to turbulent boundary layers should also be investigated.

Finally, a larger class of test cases should be studied to find any deficiencies in the wall boundary formulations. Cases that focus on phenomena such as shock wave/boundary layer interactions will further validate the ability of NASCART-GT to model these processes.

Larger Parallelization Problems

As for the parallelization enhancements made to CART3D, a study into the parallelization performance for datasets comparable to the sizes expected for viscous calculations, tens of millions of cells, should be performed. This will further validate the practicality of solving the Navier-Stokes equations on Cartesian grids. Also, investigations into ways of addressing the bandwidth limitations found in the distributed memory results might prove useful. In particular, a method of scheduling the IPC steps in order to not saturate the available bandwidth might eliminate the performance penalty associated with network collisions on an ethernet based distributed memory architecture. This research might prove useful even on shared memory architectures when very large numbers of processors are

required (say more than 2000) to solve extremely large problems that could arise with the addition of turbulence modeling in high Reynolds number three-dimensional flows.

APPENDIX A

GOVERNING EQUATIONS IN GEODESIC COORDINATES

This appendix develops the fluid dynamics equations in general curvilinear and geodesic coordinate systems. The geodesic coordinate system is first developed and is followed by a brief presentation of the governing equations in vector form. Finally, the full Navier-Stokes equations, the boundary layer equations and the Euler equations are then presented in two- and three-dimensions.

Coordinate System Basics

This section presents the basic definitions and descriptions required to develop the geodesic coordinate systems. It starts with a description of the more general curvilinear coordinate system and is followed by the geodesic coordinate system definition. Next the length elements and various curvatures are defined. Finally, all of the required vector operations are presented.

Curvilinear Coordinate System

The curvilinear coordinate system used here is simply a three-dimensional space with coordinate directions (ξ, η and ζ) that form a vector basis in the $\mathscr{R}^3$. There is no orthogonality requirement on the coordinate directions, just the following mapping requirement

$$\begin{aligned} \xi &= \xi(x,y,z) \\ \eta &= \eta(x,y,z) \\ \zeta &= \zeta(x,y,z) \end{aligned} \tag{69}$$

and the equivalent reverse mapping which holds when ξ, η and ζ form a vector basis

$$\begin{aligned} x &= x(\xi,\eta,\zeta) \\ y &= y(\xi,\eta,\zeta) \\ z &= z(\xi,\eta,\zeta) \end{aligned} \tag{70}$$

Geodesic Coordinate System

The geodesic coordinate system used here consists of a surface with coordinates, ξ and ζ, and the surface normal creating the third coordinate, η, orthogonal to ξ and ζ, see Figure 89. Notice that in general ξ, η and ζ are all functions of the Cartesian coordinate

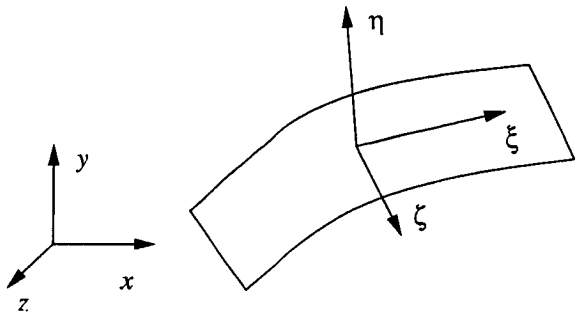


Figure 89: Example Geodesic Coordinate System

directions, x, y and z, i.e.

$$\begin{aligned} \xi &= \xi(x,y,z) \\ \eta &= \eta(x,y,z) \\ \zeta &= \zeta(x,y,z) \end{aligned} \tag{71}$$

As long as the geodesic coordinate system forms a vector basis of the Cartesian coordinate system (which it will as long as ξ and ζ are not collinear) then the following also holds

$$\begin{aligned} x &= x(\xi,\eta,\zeta) \\ y &= y(\xi,\eta,\zeta) \\ z &= z(\xi,\eta,\zeta) \end{aligned} \tag{72}$$

Differential Length Elements

A differential arc length element in the Cartesian coordinates is defined as

$$(ds)^2 = (dx)^2 + (dy)^2 + (dz)^2 \tag{73}$$

which is can also be defined in the geodesic coordinates by substituting (71) into (73) to get

$$(ds)^2 = \left(h_\xi d\xi\right)^2 + \left(h_\eta d\eta\right)^2 + \left(h_\zeta d\zeta\right)^2 \tag{74}$$

where
$$\left(h_\xi\right)^2 = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2$$
$$\left(h_\eta\right)^2 = \left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2 + \left(\frac{\partial z}{\partial \eta}\right)^2$$
$$\left(h_\zeta\right)^2 = \left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2 + \left(\frac{\partial z}{\partial \zeta}\right)^2$$

with h_ξ, h_η and h_ζ being the differential length elements in the ξ-, η- and ζ-directions, respectively.

For the curvilinear coordinate system, the differential length elements are described as

$$\begin{aligned} h_\xi &= h_\xi(\xi,\eta,\zeta) \\ h_\eta &= h_\eta(\xi,\eta,\zeta) \\ h_\zeta &= h_\zeta(\xi,\eta,\zeta) \end{aligned} \tag{75}$$

In the geodesic coordinate system, η is orthogonal to ξ and ζ, and h_η is only a function of η. Without loss of generality, h_η can be assumed to be unity. Thus, the differential length elements can be described as

$$\begin{aligned} h_\xi &= h_\xi(\xi,\eta,\zeta) \\ h_\eta &= 1 \\ h_\zeta &= h_\zeta(\xi,\eta,\zeta) \end{aligned} \tag{76}$$

Further, if the curvilinear coordinate system is only two-dimensional, then the differential length elements simplify to

$$\begin{aligned} h_\xi &= h_\xi(\xi,\eta) \\ h_\eta &= h_\eta(\xi,\eta) \\ h_\zeta &= 1 \end{aligned} \tag{77}$$

and for the two-dimensional geodesic coordinate system, then the differential length elements simplify to

$$\begin{aligned} h_\xi &= h_\xi(\xi,\eta) \\ h_\eta &= 1 \\ h_\zeta &= 1 \end{aligned} \tag{78}$$

Curvature Definitions

Three-dimensional geodesic coordinate systems have 6 curvatures that can be defined related to the differential length elements. They are expressed as K_{ab} with a being the constant coordinate for the surface and b is the coordinate direction of the curvature. For a general curvilinear coordinate system, the curvatures are defined as

$$\begin{aligned} K_{\xi\eta} &= \frac{1}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial \xi} & K_{\xi\zeta} &= \frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial \xi} \\ K_{\eta\xi} &= \frac{1}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial \eta} & K_{\eta\zeta} &= \frac{1}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial \eta} \\ K_{\zeta\xi} &= \frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial \zeta} & K_{\zeta\eta} &= \frac{1}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial \zeta} \end{aligned} \tag{79}$$

For the geodesic coordinate system, the curvatures become

$$\begin{aligned} K_{\xi\eta} &= 0 & K_{\xi\zeta} &= \frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial \xi} \\ K_{\eta\xi} &= \frac{1}{h_\xi}\frac{\partial h_\xi}{\partial \eta} & K_{\eta\zeta} &= \frac{1}{h_\zeta}\frac{\partial h_\zeta}{\partial \eta} \\ K_{\zeta\xi} &= \frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial \zeta} & K_{\zeta\eta} &= 0 \end{aligned} \tag{80}$$

For example, the first curvature in (80), $K_{\xi\eta}$, is the curvature on the constant ξ-surface in the η-direction. Notice that for this curvature, since η is independent of ξ (and ζ) in the geodesic coordinate system, this curvature is identically zero. Thus, of the six possible curvatures, only four are pertinent to this particular coordinate system.

The two-dimensional form of the curvatures is found by using (77) for the curvilinear coordinate system to get

$$\begin{aligned} K_{\xi\eta} &= \frac{1}{h_\eta}\frac{\partial h_\eta}{\partial \xi} & K_{\xi\zeta} &= 0 \\ K_{\eta\xi} &= \frac{1}{h_\xi}\frac{\partial h_\xi}{\partial \eta} & K_{\eta\zeta} &= 0 \\ K_{\zeta\xi} &= 0 & K_{\zeta\eta} &= 0 \end{aligned} \tag{81}$$

and (78) for the geodesic coordinate system to get

$$\begin{aligned} K_{\xi\eta} &= 0 & K_{\xi\zeta} &= 0 \\ K_{\eta\xi} &= \frac{1}{h_\xi}\frac{\partial h_\xi}{\partial \eta} & K_{\eta\zeta} &= 0 \\ K_{\zeta\xi} &= 0 & K_{\zeta\eta} &= 0 \end{aligned} \tag{82}$$

resulting in only the $K_{\eta\xi}$ curvature as non-zero.

Vector Operations

For a general curvilinear coordinate system, several vector operations take slightly different forms. Since the curvilinear coordinate directions may not be linearly independent, they must be included in any derivative calculation. Thus, all of the formulations utilize the following expressions for the derivative of the coordinate directions for the ξ-direction

$$\frac{\partial \bar{\imath}_\xi}{\partial \xi} = -\frac{1}{h_\eta}\frac{\partial h_\xi}{\partial \eta}\bar{\imath}_\eta - \frac{1}{h_\zeta}\frac{\partial h_\xi}{\partial \zeta}\bar{\imath}_\zeta \qquad \frac{\partial \bar{\imath}_\xi}{\partial \eta} = \frac{1}{h_\xi}\frac{\partial h_\eta}{\partial \xi}\bar{\imath}_\eta \qquad \frac{\partial \bar{\imath}_\xi}{\partial \zeta} = \frac{1}{h_\xi}\frac{\partial h_\zeta}{\partial \xi}\bar{\imath}_\zeta \tag{83}$$

the η-direction

$$\frac{\partial \bar{\imath}_\eta}{\partial \xi} = \frac{1}{h_\eta}\frac{\partial h_\xi}{\partial \eta}\bar{\imath}_\xi \qquad \frac{\partial \bar{\imath}_\eta}{\partial \eta} = -\frac{1}{h_\xi}\frac{\partial h_\eta}{\partial \xi}\bar{\imath}_\xi - \frac{1}{h_\zeta}\frac{\partial h_\eta}{\partial \zeta}\bar{\imath}_\zeta \qquad \frac{\partial \bar{\imath}_\eta}{\partial \zeta} = \frac{1}{h_\eta}\frac{\partial h_\zeta}{\partial \eta}\bar{\imath}_\zeta \tag{84}$$

and the ζ-direction

$$\frac{\partial \bar{\imath}_\zeta}{\partial \xi} = \frac{1}{h_\zeta}\frac{\partial h_\xi}{\partial \zeta}\bar{\imath}_\xi \qquad \frac{\partial \bar{\imath}_\zeta}{\partial \eta} = \frac{1}{h_\zeta}\frac{\partial h_\eta}{\partial \zeta}\bar{\imath}_\eta \qquad \frac{\partial \bar{\imath}_\zeta}{\partial \zeta} = -\frac{1}{h_\xi}\frac{\partial h_\zeta}{\partial \xi}\bar{\imath}_\xi - \frac{1}{h_\eta}\frac{\partial h_\zeta}{\partial \eta}\bar{\imath}_\eta \tag{85}$$

Gradient Operation

The gradient operation for a scalar, α, becomes

$$\nabla\alpha = \frac{1}{h_\xi}\frac{\partial \alpha}{\partial \xi}\bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial \alpha}{\partial \eta}\bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial \alpha}{\partial \zeta}\bar{\imath}_\zeta \tag{86}$$

which in two dimensions becomes

$$\nabla\alpha = \frac{1}{h_\xi}\frac{\partial \alpha}{\partial \xi}\bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial \alpha}{\partial \eta}\bar{\imath}_\eta \tag{87}$$

For the geodesic coordinate system, the gradient operation becomes

$$\nabla\alpha = \frac{1}{h_\xi}\frac{\partial \alpha}{\partial \xi}\bar{\imath}_\xi + \frac{\partial \alpha}{\partial \eta}\bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial \alpha}{\partial \zeta}\bar{\imath}_\zeta \tag{88}$$

which in two dimensions becomes

$$\nabla\alpha = \frac{1}{h_\xi}\frac{\partial\alpha}{\partial\xi}\bar{\imath}_\xi + \frac{\partial\alpha}{\partial\eta}\bar{\imath}_\eta \tag{89}$$

Divergence Operation

The divergence operation for a vector, $\boldsymbol{a}$, is found by starting with

$$\begin{aligned}\nabla\cdot\boldsymbol{a} = {} & \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial\xi} + \frac{1}{h_\eta}\frac{\partial a_\eta}{\partial\eta} + \frac{1}{h_\zeta}\frac{\partial a_\zeta}{\partial\zeta} \\ & + a_\xi\left(\frac{1}{h_\xi}\frac{\partial\bar{\imath}_\xi}{\partial\xi}\cdot\bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial\bar{\imath}_\xi}{\partial\eta}\cdot\bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial\bar{\imath}_\xi}{\partial\zeta}\cdot\bar{\imath}_\zeta\right) \\ & + a_\eta\left(\frac{1}{h_\xi}\frac{\partial\bar{\imath}_\eta}{\partial\xi}\cdot\bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial\bar{\imath}_\eta}{\partial\eta}\cdot\bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial\bar{\imath}_\eta}{\partial\zeta}\cdot\bar{\imath}_\zeta\right) \\ & + a_\zeta\left(\frac{1}{h_\xi}\frac{\partial\bar{\imath}_\zeta}{\partial\xi}\cdot\bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial\bar{\imath}_\zeta}{\partial\eta}\cdot\bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial\bar{\imath}_\zeta}{\partial\zeta}\cdot\bar{\imath}_\zeta\right)\end{aligned} \tag{90}$$

which, when the derivatives of the unit vectors are used, becomes

$$\begin{aligned}\nabla\cdot\boldsymbol{a} = {} & \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial\xi} + \frac{1}{h_\eta}\frac{\partial a_\eta}{\partial\eta} + \frac{1}{h_\zeta}\frac{\partial a_\zeta}{\partial\zeta} + \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial\xi} + \frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial\xi}\right)a_\xi \\ & + \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial\eta} + \frac{1}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial\eta}\right)a_\eta + \left(\frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial\zeta} + \frac{1}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial\zeta}\right)a_\zeta\end{aligned} \tag{91}$$

Which can be rewritten as

$$\begin{aligned}\nabla\cdot\boldsymbol{a} = {} & \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial\xi} + \frac{1}{h_\eta}\frac{\partial a_\eta}{\partial\eta} + \frac{1}{h_\zeta}\frac{\partial a_\zeta}{\partial\zeta} + \left(K_{\xi\eta} + K_{\xi\zeta}\right)a_\xi \\ & + \left(K_{\eta\xi} + K_{\eta\zeta}\right)a_\eta + \left(K_{\zeta\xi} + K_{\zeta\eta}\right)a_\zeta\end{aligned} \tag{92}$$

The two-dimensional formulation for this is

$$\nabla\cdot\boldsymbol{a} = \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial\xi} + \frac{1}{h_\eta}\frac{\partial a_\eta}{\partial\eta} + K_{\xi\eta}a_\xi + K_{\eta\xi}a_\eta \tag{93}$$

For the geodesic coordinate system, the divergence operation becomes

$$\nabla \cdot \boldsymbol{a} = \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial \xi} + \frac{\partial a_\eta}{\partial \eta} + \frac{1}{h_\zeta}\frac{\partial a_\zeta}{\partial \zeta} + K_{\xi\zeta}a_\xi + \left(K_{\eta\xi} + K_{\eta\zeta}\right)a_\eta + K_{\zeta\xi}a_\zeta \tag{94}$$

The two-dimensional formulation for this is

$$\nabla \cdot \boldsymbol{a} = \frac{1}{h_\xi}\frac{\partial a_\xi}{\partial \xi} + \frac{\partial a_\eta}{\partial \eta} + K_{\eta\xi}a_\eta \tag{95}$$

Curl Operator

The curl operator for a vector, $\boldsymbol{a}$, is found by starting with

$$\begin{aligned}
\nabla \times \boldsymbol{a} = {} & \left(\frac{1}{h_\eta}\frac{\partial a_\zeta}{\partial \eta} - \frac{1}{h_\zeta}\frac{\partial a_\eta}{\partial \zeta}\right)\bar{\imath}_\xi + \left(\frac{1}{h_\zeta}\frac{\partial a_\xi}{\partial \zeta} - \frac{1}{h_\xi}\frac{\partial a_\zeta}{\partial \xi}\right)\bar{\imath}_\eta \\
& + \left(\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial \xi} - \frac{1}{h_\eta}\frac{\partial a_\xi}{\partial \eta}\right)\bar{\imath}_\zeta \\
& + a_\xi\left(\frac{1}{h_\xi}\frac{\partial \bar{\imath}_\xi}{\partial \xi} \times \bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial \bar{\imath}_\xi}{\partial \eta} \times \bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial \bar{\imath}_\xi}{\partial \zeta} \times \bar{\imath}_\zeta\right) \\
& + a_\eta\left(\frac{1}{h_\xi}\frac{\partial \bar{\imath}_\eta}{\partial \xi} \times \bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial \bar{\imath}_\eta}{\partial \eta} \times \bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial \bar{\imath}_\eta}{\partial \zeta} \times \bar{\imath}_\zeta\right) \\
& + a_\zeta\left(\frac{1}{h_\xi}\frac{\partial \bar{\imath}_\zeta}{\partial \xi} \times \bar{\imath}_\xi + \frac{1}{h_\eta}\frac{\partial \bar{\imath}_\zeta}{\partial \eta} \times \bar{\imath}_\eta + \frac{1}{h_\zeta}\frac{\partial \bar{\imath}_\zeta}{\partial \zeta} \times \bar{\imath}_\zeta\right)
\end{aligned} \tag{96}$$

which, when the derivatives of the unit vectors are used, becomes

$$\begin{aligned}
\nabla \times \boldsymbol{a} = {} & \left[\left(\frac{1}{h_\eta}\frac{\partial a_\zeta}{\partial \eta} - \frac{1}{h_\zeta}\frac{\partial a_\eta}{\partial \zeta}\right) + \frac{1}{h_\eta h_\zeta}\left(\frac{\partial h_\zeta}{\partial \eta}a_\zeta - \frac{\partial h_\eta}{\partial \zeta}a_\eta\right)\right]\bar{\imath}_\xi \\
& + \left[\left(\frac{1}{h_\zeta}\frac{\partial a_\xi}{\partial \zeta} - \frac{1}{h_\xi}\frac{\partial a_\zeta}{\partial \xi}\right) + \frac{1}{h_\xi h_\zeta}\left(\frac{\partial h_\xi}{\partial \zeta}a_\xi - \frac{\partial h_\zeta}{\partial \xi}a_\zeta\right)\right]\bar{\imath}_\eta \\
& + \left[\left(\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial \xi} - \frac{1}{h_\eta}\frac{\partial a_\xi}{\partial \eta}\right) + \frac{1}{h_\xi h_\eta}\left(\frac{\partial h_\eta}{\partial \xi}a_\eta - \frac{\partial h_\xi}{\partial \eta}a_\xi\right)\right]\bar{\imath}_\zeta
\end{aligned} \tag{97}$$

Which can be rewritten as

$$\begin{aligned}\nabla\times\boldsymbol{a}=&\left[\left(\frac{1}{h_\eta}\frac{\partial a_\zeta}{\partial\eta}-\frac{1}{h_\zeta}\frac{\partial a_\eta}{\partial\zeta}\right)+K_{\eta\zeta}a_\zeta-K_{\zeta\eta}a_\eta\right]\bar{\imath}_\xi\\&+\left[\left(\frac{1}{h_\zeta}\frac{\partial a_\xi}{\partial\zeta}-\frac{1}{h_\xi}\frac{\partial a_\zeta}{\partial\xi}\right)+K_{\zeta\xi}a_\xi-K_{\xi\zeta}a_\zeta\right]\bar{\imath}_\eta\\&+\left[\left(\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial\xi}-\frac{1}{h_\eta}\frac{\partial a_\xi}{\partial\eta}\right)+K_{\xi\eta}a_\eta-K_{\eta\xi}a_\xi\right]\bar{\imath}_\zeta\end{aligned}\tag{98}$$

In two dimensions this is

$$\nabla\times\boldsymbol{a}=\left[\left(\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial\xi}-\frac{1}{h_\eta}\frac{\partial a_\xi}{\partial\eta}\right)+K_{\xi\eta}a_\eta-K_{\eta\xi}a_\xi\right]\bar{\imath}_\zeta\tag{99}$$

For the geodesic coordinate system, the curl operation becomes

$$\begin{aligned}\nabla\times\boldsymbol{a}=&\left[\left(\frac{\partial a_\zeta}{\partial\eta}-\frac{1}{h_\zeta}\frac{\partial a_\eta}{\partial\zeta}\right)+K_{\eta\zeta}a_\zeta\right]\bar{\imath}_\xi\\&+\left[\left(\frac{1}{h_\zeta}\frac{\partial a_\xi}{\partial\zeta}-\frac{1}{h_\xi}\frac{\partial a_\zeta}{\partial\xi}\right)+K_{\zeta\xi}a_\xi-K_{\xi\zeta}a_\zeta\right]\bar{\imath}_\eta\\&+\left[\left(\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial\xi}-\frac{\partial a_\xi}{\partial\eta}\right)-K_{\eta\xi}a_\xi\right]\bar{\imath}_\zeta\end{aligned}\tag{100}$$

For the two-dimensional geodesic coordinate system, this becomes

$$\nabla\times\boldsymbol{a}=\left[\frac{1}{h_\xi}\frac{\partial a_\eta}{\partial\xi}-\frac{\partial a_\xi}{\partial\eta}-K_{\eta\xi}a_\xi\right]\bar{\imath}_\zeta\tag{101}$$

Laplacian Operator

The Laplacian operation for a scalar, α, is combination of the gradient and divergence operators from above. Applying these operators yields

$$\begin{aligned}\nabla^2\alpha = &\frac{1}{h_\xi^2}\frac{\partial^2\alpha}{\partial\xi^2}+\frac{1}{h_\eta^2}\frac{\partial^2\alpha}{\partial\eta^2}+\frac{1}{h_\zeta^2}\frac{\partial^2\alpha}{\partial\zeta^2}\\ &+\left(\frac{1}{h_\xi}\right)\left[\frac{1}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial\xi}+\frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial\xi}-\frac{1}{h_\xi^2}\frac{\partial h_\xi}{\partial\xi}\right]\frac{\partial\alpha}{\partial\xi}\\ &+\left(\frac{1}{h_\eta}\right)\left[\frac{1}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial\eta}+\frac{1}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial\eta}-\frac{1}{h_\eta^2}\frac{\partial h_\eta}{\partial\eta}\right]\frac{\partial\alpha}{\partial\eta}\\ &+\left(\frac{1}{h_\zeta}\right)\left[\frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial\zeta}+\frac{1}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial\zeta}-\frac{1}{h_\zeta^2}\frac{\partial h_\zeta}{\partial\zeta}\right]\frac{\partial\alpha}{\partial\zeta}\end{aligned} \tag{102}$$

Which can be rewritten as

$$\begin{aligned}\nabla^2\alpha = &\frac{1}{h_\xi^2}\frac{\partial^2\alpha}{\partial\xi^2}+\frac{1}{h_\eta^2}\frac{\partial^2\alpha}{\partial\eta^2}+\frac{1}{h_\zeta^2}\frac{\partial^2\alpha}{\partial\zeta^2}\\ &+\left(\frac{1}{h_\xi}\right)\left[K_{\xi\eta}+K_{\xi\zeta}-\frac{1}{h_\xi^2}\frac{\partial h_\xi}{\partial\xi}\right]\frac{\partial\alpha}{\partial\xi}\\ &+\left(\frac{1}{h_\eta}\right)\left[K_{\eta\xi}+K_{\eta\zeta}-\frac{1}{h_\eta^2}\frac{\partial h_\eta}{\partial\eta}\right]\frac{\partial\alpha}{\partial\eta}\\ &+\left(\frac{1}{h_\zeta}\right)\left[K_{\zeta\xi}+K_{\zeta\eta}-\frac{1}{h_\zeta^2}\frac{\partial h_\zeta}{\partial\zeta}\right]\frac{\partial\alpha}{\partial\zeta}\end{aligned} \tag{103}$$

In two dimensions this is

$$\begin{aligned}\nabla^2\alpha = &\frac{1}{h_\xi^2}\frac{\partial^2\alpha}{\partial\xi^2}+\frac{1}{h_\eta^2}\frac{\partial^2\alpha}{\partial\eta^2}+\left(\frac{1}{h_\xi}\right)\left[K_{\xi\eta}-\frac{1}{h_\xi^2}\frac{\partial h_\xi}{\partial\xi}\right]\frac{\partial\alpha}{\partial\xi}\\ &+\left(\frac{1}{h_\eta}\right)\left[K_{\eta\xi}-\frac{1}{h_\eta^2}\frac{\partial h_\eta}{\partial\eta}\right]\frac{\partial\alpha}{\partial\eta}\end{aligned} \tag{104}$$

For the geodesic coordinate system, the Laplacian becomes

$$\nabla^2\alpha = \frac{1}{h_\xi^2}\frac{\partial^2\alpha}{\partial\xi^2} + \frac{\partial^2\alpha}{\partial\eta^2} + \frac{1}{h_\zeta^2}\frac{\partial^2\alpha}{\partial\zeta^2} + \left(\frac{1}{h_\xi}\right)\left[K_{\xi\zeta} - \frac{1}{h_\xi^2}\frac{\partial h_\xi}{\partial\xi}\right]\frac{\partial\alpha}{\partial\xi} \tag{105}$$
$$+ \left[K_{\eta\xi} + K_{\eta\zeta}\right]\frac{\partial\alpha}{\partial\eta} + \left(\frac{1}{h_\zeta}\right)\left[K_{\zeta\xi} - \frac{1}{h_\zeta^2}\frac{\partial h_\zeta}{\partial\zeta}\right]\frac{\partial\alpha}{\partial\zeta}$$

In two dimensions this is

$$\nabla^2\alpha = \frac{1}{h_\xi^2}\frac{\partial^2\alpha}{\partial\xi^2} + \frac{\partial^2\alpha}{\partial\eta^2} - \frac{1}{h_\xi^3}\frac{\partial h_\xi}{\partial\xi}\frac{\partial\alpha}{\partial\xi} + K_{\eta\xi}\frac{\partial\alpha}{\partial\eta} \tag{106}$$

Governing Equations in Vector Form

The most general expression of the governing equations that is independent of any coordinate system is the vector form of the governing equations. This section presents the governing equations in the vector form.

Continuity Equation

The continuity equation is simply a statement of the conservation of mass for a control volume in space. There is a balance between the density change inside the control volume and the mass flux through the control volume surfaces. In differential form this is expressed as

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = 0 \tag{107}$$

Momentum Equations

The momentum equations are a statement of Newton's Second Law of Motion for a control volume in space. This balances the momentum change within the control volume, the momentum convected through the control volume surfaces, the body forces being exerted on the control volume, the pressure gradient across the control volume and the viscous stresses applied to the control volume surfaces. In differential form this is expressed as

$$\rho\frac{\partial \boldsymbol{u}}{\partial t}+\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}=\rho\boldsymbol{f}_{\boldsymbol{body}}-\nabla p+\nabla\cdot[\tau] \tag{108}$$

where $[\tau]$ is the second order stress tensor which can be represented as

$$[\tau]=\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix}=\begin{bmatrix}\tau_{1,1} & \tau_{1,2} & \tau_{1,3}\\ \tau_{2,1} & \tau_{2,2} & \tau_{2,3}\\ \tau_{3,1} & \tau_{3,2} & \tau_{3,3}\end{bmatrix} \tag{109}$$

Energy Equations

The energy equation is an expression of the First Law of Thermodynamics for a control volume is space. This balances the energy change within the control volume, the energy convected through the control volume surfaces, the temporal change in the pressure, the temporal change in the heat production of the control volume caused by external processes, the conductive heat loss through the control volume surfaces, the work done by the body forces on the control volume, and the work done by the viscous forces. In differential form this is expressed as

$$\rho\frac{\partial H}{\partial t}+\rho\boldsymbol{u}\cdot\nabla H=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\nabla\cdot(k\nabla T)+\rho\boldsymbol{f}_{\boldsymbol{body}}\cdot\boldsymbol{u}+\nabla\cdot([\tau]\cdot\boldsymbol{u}) \tag{110}$$

Governing Equations in Geodesic Coordinates

While many researchers have developed several variations of the fluid dynamics equations in either geodesic or curvilinear coordinate systems, most have focused on the incompressible boundary layer equations in two- or three-dimensions [72, 163] with others focused on the Euler equations [140, 174] and little effort beyond [69].

Navier-Stokes Equations in Geodesic Coordinates

This section will develop the Navier-Stokes equations starting with the vector form of the Navier-Stokes equations. They will be transformed into the general curvilinear coordinate system and then the simplifications for the geodesic coordinate system will be applied to get the final form of the Navier-Stokes equations in geodesic coordinates.

Fundamental Relations

In order to simplify the derivations to follow, some fundamental relations will be developed first that will be used throughout the Navier-Stokes equation derivation.

Momentum Convection The momentum convection term starts out as

$$\begin{aligned}
\boldsymbol{u}\cdot\nabla\boldsymbol{u} = (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= \left(\frac{u_\xi}{h_\xi}\frac{\partial}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial}{\partial\zeta}\right)\boldsymbol{u} \\
&= \left(\frac{u_\xi}{h_\xi}\frac{\partial u_\xi}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\xi}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\xi}{\partial\zeta}\right)\bar{\imath}_\xi+\left(\frac{u_\xi}{h_\xi}\frac{\partial\bar{\imath}_\xi}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial\bar{\imath}_\xi}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial\bar{\imath}_\xi}{\partial\zeta}\right)u_\xi \\
&+\left(\frac{u_\xi}{h_\xi}\frac{\partial u_\eta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\eta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\eta}{\partial\zeta}\right)\bar{\imath}_\eta+\left(\frac{u_\xi}{h_\xi}\frac{\partial\bar{\imath}_\eta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial\bar{\imath}_\eta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial\bar{\imath}_\eta}{\partial\zeta}\right)u_\eta \\
&+\left(\frac{u_\xi}{h_\xi}\frac{\partial u_\zeta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\zeta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\zeta}{\partial\zeta}\right)\bar{\imath}_\zeta+\left(\frac{u_\xi}{h_\xi}\frac{\partial\bar{\imath}_\zeta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial\bar{\imath}_\zeta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial\bar{\imath}_\zeta}{\partial\zeta}\right)u_\zeta
\end{aligned} \tag{111}$$

Utilizing the derivatives of the general curvilinear coordinate system found in equations (83)–(85) this becomes

$$\begin{aligned}
\boldsymbol{u}\cdot\nabla\boldsymbol{u} &= \left[\frac{u_\xi}{h_\xi}\frac{\partial u_\xi}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\xi}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\xi}{\partial\zeta}\right]\bar{\imath}_\xi \\
&+\left[\frac{u_\xi u_\eta}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial\eta}+\frac{u_\xi u_\zeta}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial\zeta}-\frac{u_\eta^2}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial\xi}-\frac{u_\zeta^2}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial\xi}\right]\bar{\imath}_\xi \\
&+\left[\frac{u_\xi}{h_\xi}\frac{\partial u_\eta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\eta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\eta}{\partial\zeta}\right]\bar{\imath}_\eta \\
&+\left[\frac{u_\xi u_\eta}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial\xi}+\frac{u_\eta u_\zeta}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial\zeta}-\frac{u_\xi^2}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial\eta}-\frac{u_\zeta^2}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial\eta}\right]\bar{\imath}_\eta \\
&+\left[\frac{u_\xi}{h_\xi}\frac{\partial u_\zeta}{\partial\xi}+\frac{u_\eta}{h_\eta}\frac{\partial u_\zeta}{\partial\eta}+\frac{u_\zeta}{h_\zeta}\frac{\partial u_\zeta}{\partial\zeta}\right]\bar{\imath}_\zeta \\
&+\left[\frac{u_\xi u_\zeta}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial\xi}+\frac{u_\eta u_\zeta}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial\eta}-\frac{u_\xi^2}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial\zeta}-\frac{u_\eta^2}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial\zeta}\right]\bar{\imath}_\zeta
\end{aligned} \tag{112}$$

Substituting the curvature definitions this becomes

$$\begin{aligned}
\boldsymbol{u}\cdot\nabla\boldsymbol{u} = &\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial\xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial\zeta}\right]\bar{\imath}_\xi\\
&+\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\eta}u_\eta^2-K_{\xi\zeta}u_\zeta^2\right]\bar{\imath}_\xi\\
&+\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial\xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\eta}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial\zeta}\right]\bar{\imath}_\eta\\
&+\left[K_{\xi\eta}u_\xi u_\eta+K_{\zeta\eta}u_\eta u_\zeta-K_{\eta\xi}u_\xi^2-K_{\eta\zeta}u_\zeta^2\right]\bar{\imath}_\eta\\
&+\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial\xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\zeta}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial\zeta}\right]\bar{\imath}_\zeta\\
&+\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2-K_{\zeta\eta}u_\eta^2\right]\bar{\imath}_\zeta
\end{aligned}\tag{113}$$

Applying the geodesic coordinate system simplification yields

$$\begin{aligned}
\boldsymbol{u}\cdot\nabla\boldsymbol{u} = &\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial\xi}+u_\eta\frac{\partial u_\xi}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial\zeta}\right]\bar{\imath}_\xi\\
&+\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\zeta}u_\zeta^2\right]\bar{\imath}_\xi\\
&+\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial\xi}+u_\eta\frac{\partial u_\eta}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial\zeta}\right]\bar{\imath}_\eta\\
&-\left[K_{\eta\xi}u_\xi^2+K_{\eta\zeta}u_\zeta^2\right]\bar{\imath}_\eta\\
&+\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial\xi}+u_\eta\frac{\partial u_\zeta}{\partial\eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial\zeta}\right]\bar{\imath}_\zeta\\
&+\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2\right]\bar{\imath}_\zeta
\end{aligned}\tag{114}$$

Finally, in two dimensions this becomes

$$\begin{aligned}
\boldsymbol{u}\cdot\nabla\boldsymbol{u} = &\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial\xi}+u_\eta\frac{\partial u_\xi}{\partial\eta}+K_{\eta\xi}u_\xi u_\eta\right]\bar{\imath}_\xi\\
&+\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial\xi}+u_\eta\frac{\partial u_\eta}{\partial\eta}-K_{\eta\xi}u_\xi^2\right]\bar{\imath}_\eta
\end{aligned}\tag{115}$$

Stress Tensor The strain expressions in the general curvilinear coordinate system is

$$
\begin{aligned}
e_{\xi\xi} &= \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} + \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial \eta}\right)u_\eta + \left(\frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial \zeta}\right)u_\zeta \\
e_{\eta\eta} &= \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} + \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial \xi}\right)u_\xi + \left(\frac{1}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial \zeta}\right)u_\zeta \\
e_{\zeta\zeta} &= \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta} + \left(\frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial \xi}\right)u_\xi + \left(\frac{1}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial \eta}\right)u_\eta \\
e_{\xi\eta} = e_{\eta\xi} &= \left(\frac{1}{h_\xi}\right)\frac{\partial u_\eta}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta} - \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\eta}{\partial \xi}\right)u_\eta - \left(\frac{1}{h_\xi h_\eta}\frac{\partial h_\xi}{\partial \eta}\right)u_\xi \\
e_{\xi\zeta} = e_{\zeta\xi} &= \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\xi}{\partial \zeta} + \left(\frac{1}{h_\xi}\right)\frac{\partial u_\zeta}{\partial \xi} - \left(\frac{1}{h_\xi h_\zeta}\frac{\partial h_\xi}{\partial \zeta}\right)u_\xi - \left(\frac{1}{h_\xi h_\zeta}\frac{\partial h_\zeta}{\partial \xi}\right)u_\zeta \\
e_{\eta\zeta} = e_{\zeta\eta} &= \left(\frac{1}{h_\eta}\right)\frac{\partial u_\zeta}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\eta}{\partial \zeta} - \left(\frac{1}{h_\eta h_\zeta}\frac{\partial h_\eta}{\partial \zeta}\right)u_\eta - \left(\frac{1}{h_\eta h_\zeta}\frac{\partial h_\zeta}{\partial \eta}\right)u_\zeta
\end{aligned}
\tag{116}
$$

Applying the curvature definitions the strain expressions become

$$
\begin{aligned}
e_{\xi\xi} &= \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} + K_{\eta\xi}u_\eta + K_{\zeta\xi}u_\zeta \\
e_{\eta\eta} &= \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} + K_{\xi\eta}u_\xi + K_{\zeta\eta}u_\zeta \\
e_{\zeta\zeta} &= \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta} + K_{\xi\zeta}u_\xi + K_{\eta\zeta}u_\eta \\
e_{\xi\eta} &= \left(\frac{1}{h_\xi}\right)\frac{\partial u_\eta}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta} - K_{\xi\eta}u_\eta - K_{\eta\xi}u_\xi \\
e_{\xi\zeta} &= \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\xi}{\partial \zeta} + \left(\frac{1}{h_\xi}\right)\frac{\partial u_\zeta}{\partial \xi} - K_{\zeta\xi}u_\xi - K_{\xi\zeta}u_\zeta \\
e_{\eta\zeta} &= \left(\frac{1}{h_\eta}\right)\frac{\partial u_\zeta}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\eta}{\partial \zeta} - K_{\zeta\eta}u_\eta - K_{\eta\zeta}u_\zeta
\end{aligned}
\tag{117}
$$

Applying the strain relations to the stress tensor formulation results in

$$
\begin{aligned}
\tau_{\xi\xi} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\eta\xi}u_\eta + K_{\zeta\xi}u_\zeta\right) - K_{\xi\eta}u_\xi - K_{\zeta\eta}u_\zeta - K_{\xi\zeta}u_\xi - K_{\eta\zeta}u_\eta\right] \\
\tau_{\eta\eta} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\xi\eta}u_\xi + K_{\zeta\eta}u_\zeta\right) - K_{\eta\xi}u_\eta - K_{\zeta\xi}u_\zeta - K_{\xi\zeta}u_\xi - K_{\eta\zeta}u_\eta\right] \\
\tau_{\zeta\zeta} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\xi\zeta}u_\xi + K_{\eta\zeta}u_\eta\right) - K_{\eta\xi}u_\eta - K_{\zeta\xi}u_\zeta - K_{\xi\eta}u_\xi - K_{\zeta\eta}u_\zeta\right] \\
\tau_{\xi\eta} &= \mu\left[\left(\frac{1}{h_\xi}\right)\frac{\partial u_\eta}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta} - K_{\xi\eta}u_\eta - K_{\eta\xi}u_\xi\right] \\
\tau_{\xi\zeta} &= \mu\left[\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\xi}{\partial \zeta} + \left(\frac{1}{h_\xi}\right)\frac{\partial u_\zeta}{\partial \xi} - K_{\zeta\xi}u_\xi - K_{\xi\zeta}u_\zeta\right] \\
\tau_{\eta\zeta} &= \mu\left[\left(\frac{1}{h_\eta}\right)\frac{\partial u_\zeta}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\eta}{\partial \zeta} - K_{\zeta\eta}u_\eta - K_{\eta\zeta}u_\zeta\right]
\end{aligned}
\tag{118}
$$

Utilizing the geodesic coordinate system simplifications results in

$$
\begin{aligned}
\tau_{\xi\xi} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\eta\xi}u_\eta + K_{\zeta\xi}u_\zeta\right) - K_{\xi\zeta}u_\xi - K_{\eta\zeta}u_\eta\right] \\
\tau_{\eta\eta} &= \frac{2}{3}\mu\left[2\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad - \frac{2}{3}\mu\left[K_{\eta\xi}u_\eta + K_{\zeta\xi}u_\zeta + K_{\xi\zeta}u_\xi + K_{\eta\zeta}u_\eta\right] \\
\tau_{\zeta\zeta} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \frac{\partial u_\eta}{\partial \eta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\xi\zeta}u_\xi + K_{\eta\zeta}u_\eta\right) - K_{\eta\xi}u_\eta - K_{\zeta\xi}u_\zeta\right] \\
\tau_{\xi\eta} &= \mu\left[\left(\frac{1}{h_\xi}\right)\frac{\partial u_\eta}{\partial \xi} + \frac{\partial u_\xi}{\partial \eta} - K_{\eta\xi}u_\xi\right] \\
\tau_{\xi\zeta} &= \mu\left[\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\xi}{\partial \zeta} + \left(\frac{1}{h_\xi}\right)\frac{\partial u_\zeta}{\partial \xi} - K_{\zeta\xi}u_\xi - K_{\xi\zeta}u_\zeta\right] \\
\tau_{\eta\zeta} &= \mu\left[\frac{\partial u_\zeta}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\eta}{\partial \zeta} - K_{\eta\zeta}u_\zeta\right]
\end{aligned}
\tag{119}
$$

Finally, in two dimensions this becomes

$$
\begin{aligned}
\tau_{\xi\xi} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \frac{\partial u_\eta}{\partial \eta} + 2K_{\eta\xi}u_\eta\right] \\
\tau_{\eta\eta} &= \frac{2}{3}\mu\left[2\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - K_{\eta\xi}u_\eta\right] \\
\tau_{\xi\eta} &= \mu\left[\left(\frac{1}{h_\xi}\right)\frac{\partial u_\eta}{\partial \xi} + \frac{\partial u_\xi}{\partial \eta} - K_{\eta\xi}u_\xi\right]
\end{aligned}
\tag{120}
$$

Throughout the equation development in the rest of this section, the stress tensor components will take one of the above forms, depending on whether the curvilinear or geodesic formulations are being developed.

Stress Tensor Divergence The stress tensor divergence development starts with the application of the divergence operation onto the stress tensor, noting that the coordinate directions are not independent, to get

$$\nabla \cdot [\tau] = \left[\nabla \cdot \tau_\xi + \tau' \cdot \bar{\imath}_\xi\right] \bar{\imath}_\xi + \left[\nabla \cdot \tau_\eta + \tau' \cdot \bar{\imath}_\eta\right] \bar{\imath}_\eta + \left[\nabla \cdot \tau_\zeta + \tau' \cdot \bar{\imath}_\zeta\right] \bar{\imath}_\zeta \qquad (121)$$

$$\text{where} \quad \tau' = \sum_{i,j} \frac{1}{h_j} \frac{\partial \bar{\imath}_k}{\partial x_j} \tau_{jk} \quad \text{and} \quad \{i,j\} \in \{\xi, \eta, \zeta\}$$

In the stress tensor divergence expression, the first term is the divergence of the three stress tensor vectors, and the second term is the divergence of the stress tensor coordinate directions. Expanding the τ' term yields and collecting terms yields

$$\begin{aligned}
\nabla \cdot [\tau] = & \left[\frac{1}{h_\xi}\frac{\partial \tau_{\xi\xi}}{\partial \xi} + \frac{1}{h_\eta}\frac{\partial \tau_{\xi\eta}}{\partial \eta} + \frac{1}{h_\zeta}\frac{\partial \tau_{\xi\zeta}}{\partial \zeta}\right] \bar{\imath}_\xi - \left[K_{\xi\eta}\tau_{\eta\eta} + K_{\xi\zeta}\tau_{\zeta\zeta}\right] \bar{\imath}_\xi \\
& + \left[\frac{1}{h_\xi}\frac{\partial \tau_{\xi\eta}}{\partial \xi} + \frac{1}{h_\eta}\frac{\partial \tau_{\eta\eta}}{\partial \eta} + \frac{1}{h_\zeta}\frac{\partial \tau_{\eta\zeta}}{\partial \zeta}\right] \bar{\imath}_\eta - \left[K_{\eta\xi}\tau_{\xi\xi} + K_{\eta\zeta}\tau_{\zeta\zeta}\right] \bar{\imath}_\eta \\
& + \left[\frac{1}{h_\xi}\frac{\partial \tau_{\xi\zeta}}{\partial \xi} + \frac{1}{h_\eta}\frac{\partial \tau_{\eta\zeta}}{\partial \eta} + \frac{1}{h_\zeta}\frac{\partial \tau_{\eta\zeta}}{\partial \zeta}\right] \bar{\imath}_\zeta - \left[K_{\zeta\xi}\tau_{\xi\xi} + K_{\zeta\eta}\tau_{\eta\eta}\right] \bar{\imath}_\zeta \qquad (122) \\
& + \left[\left(K_{\xi\eta} + K_{\xi\zeta}\right)\tau_{\xi\xi} + \left(2K_{\eta\xi} + K_{\eta\zeta}\right)\tau_{\xi\eta} + \left(2K_{\zeta\xi} + K_{\zeta\eta}\right)\tau_{\xi\zeta}\right] \bar{\imath}_\xi \\
& + \left[\left(2K_{\xi\eta} + K_{\xi\zeta}\right)\tau_{\xi\eta} + \left(K_{\eta\xi} + K_{\eta\zeta}\right)\tau_{\eta\eta} + \left(K_{\zeta\xi} + 2K_{\zeta\eta}\right)\tau_{\eta\zeta}\right] \bar{\imath}_\eta \\
& + \left[\left(K_{\xi\eta} + 2K_{\xi\zeta}\right)\tau_{\xi\zeta} + \left(K_{\eta\xi} + 2K_{\eta\zeta}\right)\tau_{\eta\zeta} + \left(K_{\zeta\xi} + K_{\zeta\eta}\right)\tau_{\zeta\zeta}\right] \bar{\imath}_\zeta
\end{aligned}$$

For the geodesic coordinate system, this becomes

$$\begin{aligned}
\nabla\cdot[\tau] = &\left[\frac{1}{h_\xi}\frac{\partial\tau_{\xi\xi}}{\partial\xi}+\frac{\partial\tau_{\xi\eta}}{\partial\eta}+\frac{1}{h_\zeta}\frac{\partial\tau_{\xi\zeta}}{\partial\zeta}\right]\bar{\imath}_\xi \\
&+\left[\frac{1}{h_\xi}\frac{\partial\tau_{\xi\eta}}{\partial\xi}+\frac{\partial\tau_{\eta\eta}}{\partial\eta}+\frac{1}{h_\zeta}\frac{\partial\tau_{\eta\zeta}}{\partial\zeta}\right]\bar{\imath}_\eta \\
&+\left[\frac{1}{h_\xi}\frac{\partial\tau_{\xi\zeta}}{\partial\xi}+\frac{\partial\tau_{\eta\zeta}}{\partial\eta}+\frac{1}{h_\zeta}\frac{\partial\tau_{\eta\zeta}}{\partial\zeta}\right]\bar{\imath}_\zeta \\
&+\left[K_{\xi\zeta}\tau_{\xi\xi}+\left(2K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\xi\eta}+2K_{\zeta\xi}\tau_{\xi\zeta}-K_{\xi\zeta}\tau_{\zeta\zeta}\right]\bar{\imath}_\xi \\
&+\left[K_{\xi\zeta}\tau_{\xi\eta}+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\eta\eta}+K_{\zeta\xi}\tau_{\eta\zeta}-K_{\eta\xi}\tau_{\xi\xi}-K_{\eta\zeta}\tau_{\zeta\zeta}\right]\bar{\imath}_\eta \\
&+\left[2K_{\xi\zeta}\tau_{\xi\zeta}+\left(K_{\eta\xi}+2K_{\eta\zeta}\right)\tau_{\eta\zeta}+K_{\zeta\xi}\tau_{\zeta\zeta}-K_{\zeta\xi}\tau_{\xi\xi}\right]\bar{\imath}_\zeta
\end{aligned} \tag{123}$$

For the two-dimensional geodesic coordinate system, this becomes

$$\begin{aligned}
\nabla\cdot[\tau] = &\left[\frac{1}{h_\xi}\frac{\partial\tau_{\xi\xi}}{\partial\xi}+\frac{\partial\tau_{\xi\eta}}{\partial\eta}\right]\bar{\imath}_\xi+\left[\frac{1}{h_\xi}\frac{\partial\tau_{\xi\eta}}{\partial\xi}+\frac{\partial\tau_{\eta\eta}}{\partial\eta}\right]\bar{\imath}_\eta \\
&+\left[2K_{\eta\xi}\tau_{\xi\eta}\right]\bar{\imath}_\xi+\left[K_{\eta\xi}\tau_{\eta\eta}-K_{\eta\xi}\tau_{\xi\xi}\right]\bar{\imath}_\eta
\end{aligned} \tag{124}$$

Stress Tensor Energy Dissipation The stress tensor energy dissipation relation development starts with the expansion of the dot product inside the divergence operator to get

$$\nabla\cdot([\tau]\cdot\boldsymbol{u}) = \nabla\cdot\begin{bmatrix}\boldsymbol{u}\cdot\tau_\xi\\ \boldsymbol{u}\cdot\tau_\eta\\ \boldsymbol{u}\cdot\tau_\zeta\end{bmatrix} \tag{125}$$

which, after using the divergence relation for curvilinear coordinate systems, results in

$$\begin{aligned}\nabla\cdot([\tau]\cdot\boldsymbol{u}) = &\frac{1}{h_\xi}\frac{\partial}{\partial\xi}\left(u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right)\\ &+\frac{1}{h_\eta}\frac{\partial}{\partial\eta}\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right)\\ &+\frac{1}{h_\zeta}\frac{\partial}{\partial\zeta}\left(u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right)\\ &+\left(K_{\xi\eta}+K_{\xi\zeta}\right)\left(u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right)\\ &+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right)\\ &+\left(K_{\zeta\xi}+K_{\zeta\eta}\right)\left(u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right)\end{aligned} \tag{126}$$

Applying the geodesic coordinate system conditions, this becomes

$$\begin{aligned}\nabla\cdot([\tau]\cdot\boldsymbol{u}) = &\frac{1}{h_\xi}\frac{\partial}{\partial\xi}\left(u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right)\\ &+\frac{\partial}{\partial\eta}\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right)\\ &+\frac{1}{h_\zeta}\frac{\partial}{\partial\zeta}\left(u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right)\\ &+K_{\xi\zeta}\left(u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right)\\ &+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right)\\ &+K_{\zeta\xi}\left(u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right)\end{aligned} \tag{127}$$

Finally, for the two-dimensional coordinate system, this becomes

$$\begin{aligned}\nabla\cdot([\tau]\cdot\boldsymbol{u}) = &\frac{1}{h_\xi}\frac{\partial}{\partial\xi}\left(u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}\right)+\frac{\partial}{\partial\eta}\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right)\\ &+K_{\eta\xi}\left(u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right)\end{aligned} \tag{128}$$

Three-Dimensional Formulation

With all of the pieces of the three-dimensional Navier-Stokes equations developed above, they now can be assembled to complete the derivation. First, the general curvilinear coordinate system formulation will be presented, then the geodesic coordinate system will be presented for each conservation equation set.

Continuity The continuity equation uses (107) and the divergence operator equation to get

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial\left(\rho u_\eta\right)}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial\left(\rho u_\zeta\right)}{\partial \zeta} + \left(K_{\xi\eta} + K_{\xi\zeta}\right)\rho u_\xi + \left(K_{\eta\xi} + K_{\eta\zeta}\right)\rho u_\eta + \left(K_{\zeta\xi} + K_{\zeta\eta}\right)\rho u_\zeta = 0 \tag{129}$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi} + \frac{\partial\left(\rho u_\eta\right)}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial\left(\rho u_\zeta\right)}{\partial \zeta} + K_{\xi\zeta}\rho u_\xi + \left(K_{\eta\xi} + K_{\eta\zeta}\right)\rho u_\eta + K_{\zeta\xi}\rho u_\zeta = 0 \tag{130}$$

Momentum The momentum equations use (108), as well as the momentum convection and the stress tensor divergence to obtain the ξ-, η- and ζ-momentum equations. For the curvilinear coordinate formulation, the stress tensors from (118) are the appropriate ones

to be used. The ξ-momentum equation becomes

$$\begin{aligned}
&\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\eta}u_\eta^2-K_{\xi\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\xi}}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial \tau_{\xi\eta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\xi\zeta}}{\partial \zeta}\\
&\quad+\left(K_{\xi\eta}+K_{\xi\zeta}\right)\tau_{\xi\xi}+\left(2K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\xi\eta}+\left(2K_{\zeta\xi}+K_{\zeta\eta}\right)\tau_{\xi\zeta}\\
&\quad-K_{\xi\eta}\tau_{\eta\eta}-K_{\xi\zeta}\tau_{\zeta\zeta}
\end{aligned}\tag{131}$$

The η-momentum equation becomes

$$\begin{aligned}
&\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\eta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\xi\eta}u_\xi u_\eta+K_{\zeta\eta}u_\eta u_\zeta-K_{\eta\xi}u_\xi^2-K_{\eta\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\eta}}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial \tau_{\eta\eta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\eta\zeta}}{\partial \zeta}\\
&\quad+\left(2K_{\xi\eta}+K_{\xi\zeta}\right)\tau_{\xi\eta}+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\eta\eta}+\left(K_{\zeta\xi}+2K_{\zeta\eta}\right)\tau_{\eta\zeta}\\
&\quad-K_{\eta\xi}\tau_{\xi\xi}-K_{\eta\zeta}\tau_{\zeta\zeta}
\end{aligned}\tag{132}$$

The ζ-momentum equation becomes

$$\begin{aligned}
&\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2-K_{\zeta\eta}u_\eta^2\right]\\
&\quad=\rho f_{body,\zeta}-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\zeta}}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial \tau_{\eta\zeta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\zeta\zeta}}{\partial \zeta}\\
&\quad+\left(K_{\xi\eta}+2K_{\xi\zeta}\right)\tau_{\xi\zeta}+\left(K_{\eta\xi}+2K_{\eta\zeta}\right)\tau_{\eta\zeta}+\left(K_{\zeta\xi}+K_{\zeta\eta}\right)\tau_{\zeta\zeta}\\
&\quad-K_{\zeta\xi}\tau_{\xi\xi}-K_{\zeta\eta}\tau_{\eta\eta}
\end{aligned}\tag{133}$$

Applying the geodesic coordinate system simplifications and utilizing the geodesic stress tensor formulations (119) yields for the ξ-momentum equation

$$\begin{aligned}
&\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\xi}}{\partial \xi}+\frac{\partial \tau_{\xi\eta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\xi\zeta}}{\partial \zeta}\\
&\quad+K_{\xi\zeta}\tau_{\xi\xi}+\left(2K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\xi\eta}+2K_{\zeta\xi}\tau_{\xi\zeta}-K_{\xi\zeta}\tau_{\zeta\zeta}
\end{aligned}\tag{134}$$

with the η-momentum equation becoming

$$\begin{aligned}
&\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+u_\eta\frac{\partial u_\eta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial \zeta}\right]\\
&\quad-\rho\left[K_{\eta\xi}u_\xi^2+K_{\eta\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\eta}}{\partial \xi}+\frac{\partial \tau_{\eta\eta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\eta\zeta}}{\partial \zeta}\\
&\quad+K_{\xi\zeta}\tau_{\xi\eta}+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\tau_{\eta\eta}+K_{\zeta\xi}\tau_{\eta\zeta}-K_{\eta\xi}\tau_{\xi\xi}-K_{\eta\zeta}\tau_{\zeta\zeta}
\end{aligned}\tag{135}$$

and the ζ-momentum equation becoming

$$\begin{aligned}
&\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}\right]\\
&\quad\rho\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2\right]\\
&\quad=\rho f_{body,\zeta}-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\zeta}}{\partial \xi}+\frac{\partial \tau_{\eta\zeta}}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial \tau_{\zeta\zeta}}{\partial \zeta}\\
&\quad+2K_{\xi\zeta}\tau_{\xi\zeta}+\left(K_{\eta\xi}+2K_{\eta\zeta}\right)\tau_{\eta\zeta}+K_{\zeta\xi}\tau_{\zeta\zeta}-K_{\zeta\xi}\tau_{\xi\xi}-K_{\zeta\eta}\tau_{\eta\eta}
\end{aligned}\tag{136}$$

Energy The energy equation uses (110), the curvilinear vector operations and the stress tensor energy dissipation to become

$$\begin{aligned}
&\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right]\\
&\quad=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}+u_\zeta f_{body,\zeta}\right]\\
&\quad+\left(\frac{1}{h_\xi}\right)\frac{\partial}{\partial \xi}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right]\\
&\quad+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\left(\frac{1}{h_\eta}\right)k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right]\\
&\quad+\left(\frac{1}{h_\zeta}\right)\frac{\partial}{\partial \zeta}\left[\left(\frac{1}{h_\zeta}\right)k\frac{\partial T}{\partial \zeta}+u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right]\\
&\quad+\left(K_{\xi\eta}+K_{\xi\zeta}\right)\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right]\\
&\quad+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\left[\left(\frac{1}{h_\eta}\right)k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right]\\
&\quad+\left(K_{\zeta\xi}+K_{\zeta\eta}\right)\left[\left(\frac{1}{h_\zeta}\right)k\frac{\partial T}{\partial \zeta}+u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right]
\end{aligned}\tag{137}$$

Applying the geodesic coordinate system conditions, this becomes

$$\begin{aligned}
&\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right]\\
&\quad=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}+u_\zeta f_{body,\zeta}\right]\\
&\quad+\left(\frac{1}{h_\xi}\right)\frac{\partial}{\partial \xi}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right]\\
&\quad+\frac{\partial}{\partial \eta}\left[k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right]\\
&\quad+\left(\frac{1}{h_\zeta}\right)\frac{\partial}{\partial \zeta}\left[\left(\frac{1}{h_\zeta}\right)k\frac{\partial T}{\partial \zeta}+u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right]\\
&\quad+K_{\xi\zeta}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}+u_\zeta\tau_{\xi\zeta}\right]\\
&\quad+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\left[k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}+u_\zeta\tau_{\eta\zeta}\right]\\
&\quad+K_{\zeta\xi}\left[\left(\frac{1}{h_\zeta}\right)k\frac{\partial T}{\partial \zeta}+u_\xi\tau_{\xi\zeta}+u_\eta\tau_{\eta\zeta}+u_\zeta\tau_{\zeta\zeta}\right]
\end{aligned}\tag{138}$$

Two-Dimensional Formulation

With all of the pieces of the two-dimensional Navier-Stokes equations developed above, they now can be assembled to complete the derivation. First, the general curvilinear coordinate system formulation will be presented, then the geodesic coordinate system will be presented for each conservation equation set.

Continuity The continuity equation uses (107) and the divergence operator equation to get

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}+K_{\xi\eta}\rho u_\xi+K_{\eta\xi}\rho u_\eta=0 \tag{139}$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi} + \frac{\partial\left(\rho u_\eta\right)}{\partial \eta} + K_{\eta\xi}\rho u_\eta = 0 \tag{140}$$

Momentum The momentum equations use (108), as well as the momentum convection and the stress tensor divergence to obtain the ξ- and η-momentum equations. For the curvilinear coordinate formulation, the stress tensors from (118) are the appropriate ones to be used with the two-dimensional simplifications applied. The ξ-momentum equation becomes

$$\begin{aligned}\rho\frac{\partial u_\xi}{\partial t} + \rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi} + \left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta} + K_{\eta\xi}u_\xi u_\eta - K_{\xi\eta}u_\eta^2\right] \\ = \rho f_{body,\xi} - \left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi} + \left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\xi}}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial \tau_{\xi\eta}}{\partial \eta} \\ + K_{\xi\eta}\tau_{\xi\xi} + 2K_{\eta\xi}\tau_{\xi\eta} - K_{\xi\eta}\tau_{\eta\eta}\end{aligned} \tag{141}$$

The η-momentum equation becomes

$$\begin{aligned}\rho\frac{\partial u_\eta}{\partial t} + \rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi} + \left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\eta}{\partial \eta} + K_{\xi\eta}u_\xi u_\eta - K_{\eta\xi}u_\xi^2\right] \\ = \rho f_{body,\eta} - \left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta} + \left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\eta}}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial \tau_{\eta\eta}}{\partial \eta} \\ + 2K_{\xi\eta}\tau_{\xi\eta} + K_{\eta\xi}\left(\tau_{\eta\eta} - \tau_{\xi\xi}\right)\end{aligned} \tag{142}$$

Applying the geodesic coordinate system simplifications yields for the ξ-momentum equation

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}+K_{\eta\xi}u_\xi u_\eta\right] \tag{143}$$
$$=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\xi}}{\partial \xi}+\frac{\partial \tau_{\xi\eta}}{\partial \eta}+2K_{\eta\xi}\tau_{\xi\eta}$$

with the η-momentum equation becoming

$$\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+u_\eta\frac{\partial u_\eta}{\partial \eta}-K_{\eta\xi}u_\xi^2\right] \tag{144}$$
$$=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}+\left(\frac{1}{h_\xi}\right)\frac{\partial \tau_{\xi\eta}}{\partial \xi}+\frac{\partial \tau_{\eta\eta}}{\partial \eta}+K_{\eta\xi}\left(\tau_{\eta\eta}-\tau_{\xi\xi}\right)$$

Energy The energy equation uses (110), the curvilinear vector operations and the stress tensor energy dissipation to become

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}\right]$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}\right]$$
$$+\left(\frac{1}{h_\xi}\right)\frac{\partial}{\partial \xi}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}\right] \tag{145}$$
$$+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\left(\frac{1}{h_\eta}\right)k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right]$$
$$+K_{\xi\eta}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}\right]$$
$$+K_{\eta\xi}\left[\left(\frac{1}{h_\eta}\right)k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right]$$

Applying the geodesic coordinate system conditions and utilizing the geodesic stress tensor formulations (119) with the two-dimensional simplifications applied, this becomes

$$\begin{aligned}
&\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}\right] \\
&\quad=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}\right] \\
&\quad+\left(\frac{1}{h_\xi}\right)\frac{\partial}{\partial \xi}\left[\left(\frac{1}{h_\xi}\right)k\frac{\partial T}{\partial \xi}+u_\xi\tau_{\xi\xi}+u_\eta\tau_{\xi\eta}\right] \\
&\quad+\frac{\partial}{\partial \eta}\left[k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right] \\
&\quad+K_{\eta\xi}\left(k\frac{\partial T}{\partial \eta}+u_\xi\tau_{\xi\eta}+u_\eta\tau_{\eta\eta}\right)
\end{aligned} \tag{146}$$

Boundary Layer Equations in Geodesic Coordinates

The boundary layer equation will be developed from the Navier-Stokes equations and applying the standard boundary layer assumptions to the general curvilinear and geodesic formulations. In each formulation, the general curvilinear coordinate system formulations will be presented followed by the geodesic coordinate system formulations.

Assumptions

The boundary layer equations start off with the following assumptions:

1. Boundary layer thickness is small, i.e. $Re \gg 1$
2. Buoyancy effects are negligible, i.e. $Fr \gg 1$

Using these assumptions the following can be said about mathematical relations in the Navier-Stokes equations

$$\begin{aligned} u_\eta &\ll u_\xi \qquad u_\eta \ll u_\zeta \\ \frac{\partial}{\partial \eta} &\gg \frac{\partial}{\partial \xi} \qquad \frac{\partial}{\partial \eta} \gg \frac{\partial}{\partial \zeta} \\ f_{body} &\approx 0 \end{aligned} \tag{147}$$

where the first and second conditions result from assumption 1 and the third condition results from assumption 2.

In developing the boundary layer equations, an order of magnitude analysis will be done on each equation in the Navier-Stokes equation and all of the smaller terms with respect to the rest of the terms in each equation will be removed. In doing this process, the following

magnitudes are used for each group of terms in the governing equations

$$
\begin{aligned}
u_\xi &\sim \mathcal{O}(1) \\
u_\eta &\sim \mathcal{O}(\varepsilon) \\
u_\zeta &\sim \mathcal{O}(1) \\
p &\sim \mathcal{O}(1) \\
H &\sim \mathcal{O}(1) \\
\rho &\sim \mathcal{O}(1) \\
\frac{\partial}{\partial t} &\sim \mathcal{O}(1) \\
\frac{\partial}{\partial \xi} &\sim \mathcal{O}(1) \\
\frac{\partial}{\partial \eta} &\sim \mathcal{O}(1/\varepsilon) \\
\frac{\partial}{\partial \zeta} &\sim \mathcal{O}(1) \\
h_{i,j} &\sim \mathcal{O}(1) \ \{i,j\} \in \{\xi,\eta,\zeta\} \\
K_{i,j} &\sim \mathcal{O}(1) \ \{i,j\} \in \{\xi,\eta,\zeta\}
\end{aligned}
\tag{148}
$$

where $\varepsilon \ll 1$.

In preparation for the momentum and energy equation development, the shear stress components can be analyzed separately with the lowest ordered terms removed. While other terms might be removed later, it is assured that the lowest ordered terms will not remain. The stress tensor components (118) are reproduced here with the order of magnitudes

under-set each term.

$$\tau_{\xi\xi} = \frac{2}{3}\mu \left[2\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\frac{\partial u_\xi}{\partial \xi}_{[1/1]} - \underset{[1]}{\left(\frac{1}{h_\eta}\right)}\frac{\partial u_\eta}{\partial \eta}_{[\varepsilon/\varepsilon]} - \underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\frac{\partial u_\zeta}{\partial \zeta}_{[1]} \right]$$

$$+ \frac{2}{3}\mu \left[2\left(\underset{[1*\varepsilon]}{K_{\eta\xi}u_\eta} + \underset{[1*1]}{K_{\zeta\xi}u_\zeta} \right) - \underset{[1*1]}{K_{\xi\eta}u_\xi} - \underset{[1*1]}{K_{\zeta\eta}u_\zeta} - \underset{[1*1]}{K_{\xi\zeta}u_\xi} - \underset{[1*\varepsilon]}{K_{\eta\zeta}u_\eta} \right]$$

$$\tau_{\eta\eta} = \frac{2}{3}\mu \left[2\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\frac{\partial u_\eta}{\partial \eta}_{[\varepsilon/\varepsilon]} - \underset{[1]}{\left(\frac{1}{h_\xi}\right)}\frac{\partial u_\xi}{\partial \xi}_{[1/1]} - \underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\frac{\partial u_\zeta}{\partial \zeta}_{[1/1]} \right]$$

$$+ \frac{2}{3}\mu \left[2\left(\underset{[1*1]}{K_{\xi\eta}u_\xi} + \underset{[1*1]}{K_{\zeta\eta}u_\zeta} \right) - \underset{[1*\varepsilon]}{K_{\eta\xi}u_\eta} - \underset{[1*1]}{K_{\zeta\xi}u_\zeta} - \underset{[1*1]}{K_{\xi\zeta}u_\xi} - \underset{[1*\varepsilon]}{K_{\eta\zeta}u_\eta} \right]$$

$$\tau_{\zeta\zeta} = \frac{2}{3}\mu \left[2\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\frac{\partial u_\zeta}{\partial \zeta}_{[1/1]} - \underset{[1]}{\left(\frac{1}{h_\xi}\right)}\frac{\partial u_\xi}{\partial \xi}_{[1/1]} - \underset{[1]}{\left(\frac{1}{h_\eta}\right)}\frac{\partial u_\eta}{\partial \eta}_{[\varepsilon/\varepsilon]} \right] \tag{149}$$

$$+ \frac{2}{3}\mu \left[2\left(\underset{[1*1]}{K_{\xi\zeta}u_\xi} + \underset{[1*\varepsilon]}{K_{\eta\zeta}u_\eta} \right) - \underset{[1*\varepsilon]}{K_{\eta\xi}u_\eta} - \underset{[1*1]}{K_{\zeta\xi}u_\zeta} - \underset{[1*1]}{K_{\xi\eta}u_\xi} - \underset{[1*1]}{K_{\zeta\eta}u_\zeta} \right]$$

$$\tau_{\xi\eta} = \mu \left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\frac{\partial u_\eta}{\partial \xi}_{[\varepsilon/1]} + \underset{[1]}{\left(\frac{1}{h_\eta}\right)}\frac{\partial u_\xi}{\partial \eta}_{[1/\varepsilon]} - \underset{[1*\varepsilon]}{K_{\xi\eta}u_\eta} - \underset{[1*1]}{K_{\eta\xi}u_\xi} \right]$$

$$\tau_{\xi\zeta} = \mu \left[\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\frac{\partial u_\xi}{\partial \zeta}_{[1/1]} + \underset{[1]}{\left(\frac{1}{h_\xi}\right)}\frac{\partial u_\zeta}{\partial \xi}_{[1/1]} - \underset{[1*1]}{K_{\zeta\xi}u_\xi} - \underset{[1*1]}{K_{\xi\zeta}u_\zeta} \right]$$

$$\tau_{\eta\zeta} = \mu \left[\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\frac{\partial u_\zeta}{\partial \eta}_{[1/\varepsilon]} + \underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\frac{\partial u_\eta}{\partial \zeta}_{[\varepsilon/1]} - \underset{[1*\varepsilon]}{K_{\zeta\eta}u_\eta} - \underset{[1*1]}{K_{\eta\zeta}u_\zeta} \right]$$

It is clear that all terms of order $\mathcal{O}(\varepsilon)$ can be ignored, which results in

$$
\begin{aligned}
\tau_{\xi\xi} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad + \frac{2}{3}\mu\left[2K_{\zeta\xi}u_\zeta - K_{\xi\eta}u_\xi - K_{\zeta\eta}u_\zeta - K_{\xi\zeta}u_\xi\right] \sim \mathcal{O}(1) \\
\tau_{\eta\eta} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta}\right] \\
&\quad + \frac{2}{3}\mu\left[2\left(K_{\xi\eta}u_\xi + K_{\zeta\eta}u_\zeta\right) - K_{\zeta\xi}u_\zeta - K_{\xi\zeta}u_\xi\right] \sim \mathcal{O}(1) \\
\tau_{\zeta\zeta} &= \frac{2}{3}\mu\left[2\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\zeta}{\partial \zeta} - \left(\frac{1}{h_\xi}\right)\frac{\partial u_\xi}{\partial \xi} - \left(\frac{1}{h_\eta}\right)\frac{\partial u_\eta}{\partial \eta}\right] \\
&\quad + \frac{2}{3}\mu\left[2K_{\xi\zeta}u_\xi - K_{\zeta\xi}u_\zeta - K_{\xi\eta}u_\xi - K_{\zeta\eta}u_\zeta\right] \sim \mathcal{O}(1) \\
\tau_{\xi\eta} &= \mu\left[\left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta} - K_{\eta\xi}u_\xi\right] \sim \mathcal{O}(1/\varepsilon) \\
\tau_{\xi\zeta} &= \mu\left[\left(\frac{1}{h_\zeta}\right)\frac{\partial u_\xi}{\partial \zeta} + \left(\frac{1}{h_\xi}\right)\frac{\partial u_\zeta}{\partial \xi} - K_{\zeta\xi}u_\xi - K_{\xi\zeta}u_\zeta\right] \sim \mathcal{O}(1) \\
\tau_{\eta\zeta} &= \mu\left[\left(\frac{1}{h_\eta}\right)\frac{\partial u_\zeta}{\partial \eta} - K_{\eta\zeta}u_\zeta\right] \sim \mathcal{O}(1/\varepsilon)
\end{aligned}
\tag{150}
$$

Three-Dimensional Formulation

The three-dimensional formulation of the boundary layer equations starts with the Navier-Stokes equations and then applies the boundary layer assumptions described above. Each conservation equation set will first develop the curvilinear boundary layer equations and then the geodesic coordinate system equations will be developed.

Continuity The continuity equation starts with the Navier-Stokes continuity equation (129) reproduced here with the order of magnitudes under-set each term.

$$\underset{[1]}{\frac{\partial \rho}{\partial t}} + \underset{[1]}{\left(\frac{1}{h_\xi}\right)} \underset{[1/1]}{\frac{\partial \left(\rho u_\xi\right)}{\partial \xi}} + \underset{[1]}{\left(\frac{1}{h_\eta}\right)} \underset{[\varepsilon/\varepsilon]}{\frac{\partial \left(\rho u_\eta\right)}{\partial \eta}} + \underset{[1]}{\left(\frac{1}{h_\zeta}\right)} \underset{[1]}{\frac{\partial \left(\rho u_\zeta\right)}{\partial \zeta}} \tag{151}$$
$$+ \underset{[1]}{\left(K_{\xi\eta} + K_{\xi\zeta}\right)} \underset{[1]}{\rho u_\xi} + \underset{[1]}{\left(K_{\eta\xi} + K_{\eta\zeta}\right)} \underset{[\varepsilon]}{\rho u_\eta} + \underset{[1]}{\left(K_{\zeta\xi} + K_{\zeta\eta}\right)} \underset{[1]}{\rho u_\zeta} = 0$$

Thus, the terms of order $\mathcal{O}(\varepsilon)$ can be eliminated which results in the following

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right) \frac{\partial \left(\rho u_\xi\right)}{\partial \xi} + \left(\frac{1}{h_\eta}\right) \frac{\partial \left(\rho u_\eta\right)}{\partial \eta} + \left(\frac{1}{h_\zeta}\right) \frac{\partial \left(\rho u_\zeta\right)}{\partial \zeta} \tag{152}$$
$$+ \left(K_{\xi\eta} + K_{\xi\zeta}\right) \rho u_\xi + \left(K_{\zeta\xi} + K_{\zeta\eta}\right) \rho u_\zeta = 0$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right) \frac{\partial \left(\rho u_\xi\right)}{\partial \xi} + \frac{\partial \left(\rho u_\eta\right)}{\partial \eta} + \left(\frac{1}{h_\zeta}\right) \frac{\partial \left(\rho u_\zeta\right)}{\partial \zeta} + K_{\xi\zeta} \rho u_\xi + K_{\zeta\xi} \rho u_\zeta = 0 \tag{153}$$

Momentum In order to simplify the order of magnitude analysis, all shear stress components are first assumed to be the order developed above and then each remaining shear stress component will be included into the equations and any further eliminations needed can then be done. The shear stress terms will be evaluated separately from the rest of the terms in the momentum equations in order to retain both the shear stress and convective contributions. The ξ-momentum equation (131) is reproduced here with the order of

magnitudes under-set each term.

$$
\begin{aligned}
&\underset{[1/1]}{\rho\frac{\partial u_\xi}{\partial t}}+\rho\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1*1/1]}{u_\xi\frac{\partial u_\xi}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[\varepsilon*1/\varepsilon]}{u_\eta\frac{\partial u_\xi}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1*1/1]}{u_\zeta\frac{\partial u_\xi}{\partial \zeta}}\right]\\
&\quad+\rho\left[\underset{[1*1*\varepsilon]}{K_{\eta\xi}u_\xi u_\eta}+\underset{[1*1*1]}{K_{\zeta\xi}u_\xi u_\zeta}-\underset{[1*\varepsilon^2]}{K_{\xi\eta}u_\eta^2}-\underset{[1*1^2]}{K_{\xi\zeta}u_\zeta^2}\right]\\
&\quad=\underset{[0]}{\rho f_{body,\xi}}-\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1/1]}{\frac{\partial p}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1/1]}{\frac{\partial \tau_{\xi\xi}}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[(1/\varepsilon)/\varepsilon]}{\frac{\partial \tau_{\xi\eta}}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1/1]}{\frac{\partial \tau_{\xi\zeta}}{\partial \zeta}}\\
&\quad+\underset{[1]}{\left(K_{\xi\eta}+K_{\xi\zeta}\right)}\underset{[1]}{\tau_{\xi\xi}}+\underset{[1]}{\left(2K_{\eta\xi}+K_{\eta\zeta}\right)}\underset{[1/\varepsilon]}{\tau_{\xi\eta}}+\underset{[1]}{\left(2K_{\zeta\xi}+K_{\zeta\eta}\right)}\underset{[1]}{\tau_{\xi\zeta}}\\
&\quad-\underset{[1*1]}{K_{\xi\eta}\tau_{\eta\eta}}-\underset{[1*1]}{K_{\xi\zeta}\tau_{\zeta\zeta}}
\end{aligned}
\tag{154}
$$

For the shear stress terms, terms below order $\mathcal{O}(1/\varepsilon^2)$ can be ignored, while for the rest of the terms, terms below order $\mathcal{O}(1)$ can be eliminated. Notice that the $\tau_{\xi\eta}$ term is the only remaining shear stress term, and recall that the only $\mathcal{O}(1/\varepsilon)$ term in that shear stress term is the $\partial u_\xi/\partial\eta$ term. Thus the ξ-momentum equation for the curvilinear coordinate system becomes

$$
\begin{aligned}
&\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\zeta}u_\zeta^2\right]\\
&\quad=-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\mu\left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta}\right]
\end{aligned}
\tag{155}
$$

which, when incorporating the geodesic coordinate system simplifications, becomes

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\zeta}u_\zeta^2\right] \tag{156}$$
$$=-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\frac{\partial}{\partial \eta}\left(\mu\frac{\partial u_\xi}{\partial \eta}\right)$$

The η-momentum equation (132) is reproduced here with the order of magnitudes underset each term.

$$\underset{[1/1]}{\rho\frac{\partial u_\eta}{\partial t}}+\rho\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1*\varepsilon/1]}{u_\xi\frac{\partial u_\eta}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[\varepsilon*\varepsilon/\varepsilon]}{u_\eta\frac{\partial u_\eta}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1*\varepsilon/1]}{u_\zeta\frac{\partial u_\eta}{\partial \zeta}}\right]$$
$$+\rho\left[\underset{[1*1*\varepsilon]}{K_{\xi\eta}u_\xi u_\eta}+\underset{[1*\varepsilon*1]}{K_{\zeta\eta}u_\eta u_\zeta}-\underset{[1*1^2]}{K_{\eta\xi}u_\xi^2}-\underset{[1*1^2]}{K_{\eta\zeta}u_\zeta^2}\right]$$
$$=\underset{[0]}{\rho f_{body,\eta}}-\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[1/\varepsilon]}{\frac{\partial p}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[(1/\varepsilon)/1]}{\frac{\partial \tau_{\xi\eta}}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[1/\varepsilon]}{\frac{\partial \tau_{\eta\eta}}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[(1/\varepsilon)/1]}{\frac{\partial \tau_{\eta\zeta}}{\partial \zeta}} \tag{157}$$
$$+\underset{[1]}{\left(2K_{\xi\eta}+K_{\xi\zeta}\right)}\underset{[1/\varepsilon]}{\tau_{\xi\eta}}+\underset{[1]}{\left(K_{\eta\xi}+K_{\eta\zeta}\right)}\underset{[1]}{\tau_{\eta\eta}}+\underset{[1]}{\left(K_{\zeta\xi}+2K_{\zeta\eta}\right)}\underset{[1/\varepsilon]}{\tau_{\eta\zeta}}$$
$$-\underset{[1*1]}{K_{\eta\xi}\tau_{\xi\xi}}-\underset{[1*1]}{K_{\eta\zeta}\tau_{\zeta\zeta}}$$

To maintain consistency with the ξ-direction momentum development above, the shear stress terms below order $\mathcal{O}\left(1/\varepsilon^2\right)$ are ignored, while for the rest of the terms in the η-momentum equation, terms below order $\mathcal{O}(1)$ can be eliminated. What results is

$$\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}=\rho\left[K_{\eta\xi}u_\xi^2+K_{\eta\zeta}u_\zeta^2\right] \tag{158}$$

and the formulation for the geodesic coordinate system is

$$\frac{\partial p}{\partial \eta}=\rho\left[K_{\eta\xi}u_\xi^2+K_{\eta\zeta}u_\zeta^2\right] \tag{159}$$

The ζ-momentum equation (133) is reproduced here with the order of magnitudes underset each term.

$$\begin{aligned}
&\underset{[1/1]}{\rho\frac{\partial u_\zeta}{\partial t}}+\rho\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1*1/1]}{u_\xi\frac{\partial u_\zeta}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[\varepsilon*1/\varepsilon]}{u_\eta\frac{\partial u_\zeta}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1*1/1]}{u_\zeta\frac{\partial u_\zeta}{\partial \zeta}}\right]\\
&\quad+\rho\left[\underset{[1*1*1]}{K_{\xi\zeta}u_\xi u_\zeta}+\underset{[1*\varepsilon*1]}{K_{\eta\zeta}u_\eta u_\zeta}-\underset{[1*1^2]}{K_{\zeta\xi}u_\xi^2}-\underset{[1*\varepsilon^2]}{K_{\zeta\eta}u_\eta^2}\right]\\
&\quad=\underset{[0]}{\rho f_{body,\zeta}}-\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1/1]}{\frac{\partial p}{\partial \zeta}}+\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1/1]}{\frac{\partial \tau_{\xi\zeta}}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[(1/\varepsilon)/\varepsilon]}{\frac{\partial \tau_{\eta\zeta}}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1/1]}{\frac{\partial \tau_{\zeta\zeta}}{\partial \zeta}}\\
&\quad+\underset{[1]}{\left(K_{\xi\eta}+2K_{\xi\zeta}\right)}\underset{[1]}{\tau_{\xi\zeta}}+\underset{[1]}{\left(K_{\eta\xi}+2K_{\eta\zeta}\right)}\underset{[1/\varepsilon]}{\tau_{\eta\zeta}}+\underset{[1]}{\left(K_{\zeta\xi}+K_{\zeta\eta}\right)}\underset{[1]}{\tau_{\zeta\zeta}}\\
&\quad-\underset{[1*1]}{K_{\zeta\xi}\tau_{\xi\xi}}-\underset{[1*1]}{K_{\zeta\eta}\tau_{\eta\eta}}
\end{aligned}\tag{160}$$

To maintain consistency with the ξ-direction momentum development above, the shear stress terms below order $\mathcal{O}(1/\varepsilon^2)$ are ignored, while for the rest of the terms in the η-momentum equation, terms below order $\mathcal{O}(1)$ can be eliminated. Notice, as above, that the $\tau_{\eta\zeta}$ term is the only remaining shear stress term, and recall that the only $\mathcal{O}(1/\varepsilon)$ term in that shear stress term is the $\partial u_\zeta/\partial\eta$ term. Thus the ζ-momentum equation for the curvilinear coordinate system becomes

$$\begin{aligned}
&\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}\right]\\
&\quad\rho\left[K_{\xi\zeta}u_\xi u_\zeta-K_{\zeta\xi}u_\xi^2\right]\\
&\quad=-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\mu\left(\frac{1}{h_\eta}\right)\frac{\partial u_\zeta}{\partial \eta}\right]
\end{aligned}\tag{161}$$

which, when incorporating the geodesic coordinate system simplifications, becomes

$$\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}+K_{\xi\zeta}u_\xi u_\zeta-K_{\zeta\xi}u_\xi^2\right] \qquad (162)$$
$$=-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}+\frac{\partial}{\partial \eta}\left(\mu\frac{\partial u_\zeta}{\partial \eta}\right)$$

Energy Similar to the momentum development, all shear stress components are first assumed to be the order developed above and then each remaining shear stress component will be included into the equations and any further eliminations needed can then be done. The energy equation (137) is reproduced here with the order of magnitudes under-set each

term.

$$
\begin{aligned}
&\underset{[1/1]}{\rho\frac{\partial H}{\partial t}}+\rho\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1*1/1]}{u_\xi\frac{\partial H}{\partial \xi}}+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[\varepsilon*1/\varepsilon]}{u_\eta\frac{\partial H}{\partial \eta}}+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1*1/1]}{u_\zeta\frac{\partial H}{\partial \zeta}}\right] \\
&\quad=\underset{[1/1]}{\frac{\partial p}{\partial t}}+\underset{[1/1]}{\frac{\partial Q}{\partial t}}+\rho\left[\underset{[1*0]}{u_\xi f_{body,\xi}}+\underset{[1*0]}{u_\eta f_{body,\eta}}+\underset{[1*0]}{u_\zeta f_{body,\zeta}}\right] \\
&\quad+\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1]}{\frac{\partial}{\partial \xi}}\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1/1]}{k\frac{\partial T}{\partial \xi}}+\underset{[1*1]}{u_\xi\tau_{\xi\xi}}+\underset{[\varepsilon*(1/\varepsilon)]}{u_\eta\tau_{\xi\eta}}+\underset{[1*1]}{u_\zeta\tau_{\xi\zeta}}\right] \\
&\quad+\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[1/\varepsilon]}{\frac{\partial}{\partial \eta}}\left[\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[1/\varepsilon]}{k\frac{\partial T}{\partial \eta}}+\underset{[1*(1/\varepsilon)]}{u_\xi\tau_{\xi\eta}}+\underset{[\varepsilon*1]}{u_\eta\tau_{\eta\eta}}+\underset{[1*(1/\varepsilon)]}{u_\zeta\tau_{\eta\zeta}}\right] \\
&\quad+\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1]}{\frac{\partial}{\partial \zeta}}\left[\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1/1]}{k\frac{\partial T}{\partial \zeta}}+\underset{[1*1]}{u_\xi\tau_{\xi\zeta}}+\underset{[\varepsilon*(1/\varepsilon)]}{u_\eta\tau_{\eta\zeta}}+\underset{[1*1]}{u_\zeta\tau_{\zeta\zeta}}\right] \\
&\quad+\left(\underset{[1]}{K_{\xi\eta}}+\underset{[1]}{K_{\xi\zeta}}\right)\left[\underset{[1]}{\left(\frac{1}{h_\xi}\right)}\underset{[1/1]}{k\frac{\partial T}{\partial \xi}}+\underset{[1*1]}{u_\xi\tau_{\xi\xi}}+\underset{[\varepsilon*(1/\varepsilon)]}{u_\eta\tau_{\xi\eta}}+\underset{[1*1]}{u_\zeta\tau_{\xi\zeta}}\right] \\
&\quad+\left(\underset{[1]}{K_{\eta\xi}}+\underset{[1]}{K_{\eta\zeta}}\right)\left[\underset{[1]}{\left(\frac{1}{h_\eta}\right)}\underset{[1/\varepsilon]}{k\frac{\partial T}{\partial \eta}}+\underset{[1*(1/\varepsilon)]}{u_\xi\tau_{\xi\eta}}+\underset{[1*1]}{u_\eta\tau_{\eta\eta}}+\underset{[1*(1/\varepsilon)]}{u_\zeta\tau_{\eta\zeta}}\right] \\
&\quad+\left(\underset{[1]}{K_{\zeta\xi}}+\underset{[1]}{K_{\zeta\eta}}\right)\left[\underset{[1]}{\left(\frac{1}{h_\zeta}\right)}\underset{[1/1]}{k\frac{\partial T}{\partial \zeta}}+\underset{[1*1]}{u_\xi\tau_{\xi\zeta}}+\underset{[\varepsilon*(1/\varepsilon)]}{u_\eta\tau_{\eta\zeta}}+\underset{[1*1]}{u_\zeta\tau_{\zeta\zeta}}\right]
\end{aligned}
\tag{163}
$$

For the shear stress and thermal conductivity terms, terms below order $\mathcal{O}(1/\varepsilon^2)$ can be ignored, while for the rest of the terms, terms below order $\mathcal{O}(1)$ can be eliminated. Notice that the $\tau_{\xi\eta}$ and $\tau_{\zeta\eta}$ terms are the only remaining shear stress terms, and recall that the only $\mathcal{O}(1/\varepsilon)$ terms in these shear stress terms are the $\partial/\partial\eta$ terms. Thus the energy equation

for the curvilinear coordinate system becomes

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right]$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\left(\frac{1}{h_\eta}\right)k\frac{\partial T}{\partial \eta}\right] \tag{164}$$
$$+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\mu\left(\frac{1}{h_\eta}\right)u_\xi\frac{\partial u_\xi}{\partial \eta}+\mu\left(\frac{1}{h_\eta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right]$$

To proceed, the conductivity term is converted to terms of the stagnation enthalpy and velocities to get (after an order of magnitude analysis eliminates the u_η term)

$$k\left(\frac{1}{h_\eta}\right)\frac{\partial T}{\partial \eta}=\left(\frac{\mu}{Pr}\right)\left(\frac{1}{h_\eta}\right)\left[\frac{\partial H}{\partial \eta}-u_\xi\frac{\partial u_\xi}{\partial \eta}-u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right] \tag{165}$$

Also, the shear stress components can be manipulated to get the following if the viscosity gradient is assumed to be $\mathcal{O}(\varepsilon)$

$$\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\mu\left(\frac{1}{h_\eta}\right)u_\xi\frac{\partial u_\xi}{\partial \eta}+\mu\left(\frac{1}{h_\eta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right] \tag{166}$$
$$=\mu\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\left(\frac{1}{h_\eta}\right)u_\xi\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\eta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right]$$

Combining these two results with the curvilinear energy equation formulation results in

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right]$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t} \tag{167}$$
$$+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\left(\frac{1}{h_\eta}\right)\frac{\partial H}{\partial \eta}+\mu\left(1-\frac{1}{Pr}\right)\left(\frac{1}{h_\eta}\right)\left(u_\xi\frac{\partial u_\xi}{\partial \eta}+u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right)\right]$$

which, when incorporating the geodesic coordinate system simplifications, becomes

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right] \tag{168}$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\frac{\partial H}{\partial \eta}+\mu\left(1-\frac{1}{Pr}\right)\left(u_\xi\frac{\partial u_\xi}{\partial \eta}+u_\zeta\frac{\partial u_\zeta}{\partial \eta}\right)\right]$$

Two-Dimensional Formulation

The development of the two-dimensional formulations of the boundary layer equations follows the same path as the three-dimensional formulation, with the removal of the third coordinate direction.

Continuity The boundary layer continuity equation in the general curvilinear coordinate system becomes

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}+K_{\xi\eta}\rho u_\xi=0 \tag{169}$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}=0 \tag{170}$$

Momentum The boundary layer ξ-momentum equation in the general curvilinear coordinate system becomes

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta}\right] \tag{171}$$
$$=-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\mu\left(\frac{1}{h_\eta}\right)\frac{\partial u_\xi}{\partial \eta}\right]$$

In the geodesic coordinate system this becomes

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}\right]=-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}+\frac{\partial}{\partial \eta}\left(\mu\frac{\partial u_\xi}{\partial \eta}\right) \tag{172}$$

The boundary layer η-momentum equation in the general curvilinear coordinate system becomes

$$\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}=\rho K_{\eta\xi}u_\xi^2 \tag{173}$$

In the geodesic coordinate system this becomes

$$\frac{\partial p}{\partial \eta}=\rho K_{\eta\xi}u_\xi^2 \tag{174}$$

Energy The boundary layer energy equation in the general curvilinear coordinate system becomes

$$\begin{aligned}\rho\frac{\partial H}{\partial t}+\rho&\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}\right]\\&=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}\\&+\left(\frac{1}{h_\eta}\right)\frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\left(\frac{1}{h_\eta}\right)\frac{\partial H}{\partial \eta}+\mu\left(1-\frac{1}{Pr}\right)\left(\frac{1}{h_\eta}\right)u_\xi\frac{\partial u_\xi}{\partial \eta}\right]\end{aligned} \tag{175}$$

In the geodesic coordinate system this becomes

$$\begin{aligned}\rho\frac{\partial H}{\partial t}+\rho&\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}\right]\\&=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\frac{\partial}{\partial \eta}\left[\frac{\mu}{Pr}\frac{\partial H}{\partial \eta}+\mu\left(1-\frac{1}{Pr}\right)u_\xi\frac{\partial u_\xi}{\partial \eta}\right]\end{aligned} \tag{176}$$

Euler Equations in Geodesic Coordinates

This section will develop the Euler equations from the vector of the Navier-Stokes equations from above applying the requisite assumptions. First the general curvilinear

coordinate system form will be presented, followed by the geodesic coordinate system form.

Assumptions

The primary differences between the Navier-Stokes and the Euler equations are the assumptions of an inviscid and adiabatic flow. The first results in the viscosity, μ, to approach zero, and the second results in the thermal conductivity, k, to approach zero and no heat production caused by external processes, $\partial Q/\partial t \approx 0$.

Three-Dimensional Formulation

The three-dimensional formulations of the Euler equations follow a similar development as the Navier-Stokes equations developed above. The major difference is the added simplifications that can be made with respect to the inviscid and adiabatic assumptions. First, the general curvilinear coordinate system formulation will be presented, then the geodesic coordinate system formulation will be presented for each conservation equation set.

Continuity The continuity equation uses (107) and the divergence operator equation to get

$$\frac{\partial \rho}{\partial t} + \left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi} + \left(\frac{1}{h_\eta}\right)\frac{\partial\left(\rho u_\eta\right)}{\partial \eta} + \left(\frac{1}{h_\zeta}\right)\frac{\partial\left(\rho u_\zeta\right)}{\partial \zeta} \tag{177}$$
$$+ \left(K_{\xi\eta} + K_{\xi\zeta}\right)\rho u_\xi + \left(K_{\eta\xi} + K_{\eta\zeta}\right)\rho u_\eta + \left(K_{\zeta\xi} + K_{\zeta\eta}\right)\rho u_\zeta = 0$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)\frac{\partial\left(\rho u_\zeta\right)}{\partial \zeta} \tag{178}$$
$$+K_{\xi\zeta}\rho u_\xi+\left(K_{\eta\xi}+K_{\eta\zeta}\right)\rho u_\eta+K_{\zeta\xi}\rho u_\zeta=0$$

Momentum The momentum equations use (108), as well as the momentum convection to obtain the ξ-, η- and ζ-momentum equations. Notice that the stress tensor divergence is not needed since it only contains viscous terms. The ξ-momentum equation becomes

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}\right]$$
$$+\rho\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\eta}u_\eta^2-K_{\xi\zeta}u_\zeta^2\right] \tag{179}$$
$$=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}$$

The η-momentum equation becomes

$$\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\eta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial \zeta}\right]$$
$$+\rho\left[K_{\xi\eta}u_\xi u_\eta+K_{\zeta\eta}u_\eta u_\zeta-K_{\eta\xi}u_\xi^2-K_{\eta\zeta}u_\zeta^2\right] \tag{180}$$
$$=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}$$

The ζ-momentum equation becomes

$$\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}\right]$$
$$+\rho\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2-K_{\zeta\eta}u_\eta^2\right] \tag{181}$$
$$=\rho f_{body,\zeta}-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}$$

Applying the geodesic coordinate system simplifications yields for the ξ-momentum equation

$$\begin{aligned}
&\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\xi}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\eta\xi}u_\xi u_\eta+K_{\zeta\xi}u_\xi u_\zeta-K_{\xi\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}
\end{aligned}\tag{182}$$

with the η-momentum equation becoming

$$\begin{aligned}
&\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+u_\eta\frac{\partial u_\eta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\eta}{\partial \zeta}\right]\\
&\quad-\rho\left[K_{\eta\xi}u_\xi^2+K_{\eta\zeta}u_\zeta^2\right]\\
&\quad=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}
\end{aligned}\tag{183}$$

and the ζ-momentum equation becoming

$$\begin{aligned}
&\rho\frac{\partial u_\zeta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\zeta}{\partial \xi}+u_\eta\frac{\partial u_\zeta}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial u_\zeta}{\partial \zeta}\right]\\
&\quad+\rho\left[K_{\xi\zeta}u_\xi u_\zeta+K_{\eta\zeta}u_\eta u_\zeta-K_{\zeta\xi}u_\xi^2\right]\\
&\quad=\rho f_{body,\zeta}-\left(\frac{1}{h_\zeta}\right)\frac{\partial p}{\partial \zeta}
\end{aligned}\tag{184}$$

Energy The energy equation uses (110) and the curvilinear vector operations, notice that the stress tensor energy dissipation as well as the heat production and conduction terms disappear due to the assumptions of inviscid and adiabatic, to become

$$\begin{aligned}
&\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right]\\
&\quad=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}+u_\zeta f_{body,\zeta}\right]
\end{aligned}\tag{185}$$

Applying the geodesic coordinate system conditions, this becomes

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}+\left(\frac{1}{h_\zeta}\right)u_\zeta\frac{\partial H}{\partial \zeta}\right] \tag{186}$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}+u_\zeta f_{body,\zeta}\right]$$

Two-Dimensional Formulation

The two-dimensional formulations of the Euler equations follow a similar development as the Navier-Stokes equations developed above. The major difference is the added simplifications that can be made with respect to the inviscid and adiabatic assumptions. First, the general curvilinear coordinate system formulation will be presented, then the geodesic coordinate system formulation will be presented for each conservation equation set.

Continuity The continuity equation uses (107) and the divergence operator equation to get

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\left(\frac{1}{h_\eta}\right)\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}+K_{\xi\eta}\rho u_\xi+K_{\eta\xi}\rho u_\eta=0 \tag{187}$$

In the geodesic coordinate system this becomes

$$\frac{\partial \rho}{\partial t}+\left(\frac{1}{h_\xi}\right)\frac{\partial\left(\rho u_\xi\right)}{\partial \xi}+\frac{\partial\left(\rho u_\eta\right)}{\partial \eta}+K_{\eta\xi}\rho u_\eta=0 \tag{188}$$

Momentum The momentum equations use (108), as well as the momentum convection to obtain the ξ- and η-momentum equations. Notice that the stress tensor divergence is not

needed since it only contains viscous terms. The ξ-momentum equation becomes

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\xi}{\partial \eta}+K_{\eta\xi}u_\xi u_\eta-K_{\xi\eta}u_\eta^2\right] \tag{189}$$
$$=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi}$$

The η-momentum equation becomes

$$\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial u_\eta}{\partial \eta}+K_{\xi\eta}u_\xi u_\eta-K_{\eta\xi}u_\xi^2\right] \tag{190}$$
$$=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta}$$

Applying the geodesic coordinate system simplifications yields for the ξ-momentum equation

$$\rho\frac{\partial u_\xi}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\xi}{\partial \xi}+u_\eta\frac{\partial u_\xi}{\partial \eta}+K_{\eta\xi}u_\xi u_\eta\right]=\rho f_{body,\xi}-\left(\frac{1}{h_\xi}\right)\frac{\partial p}{\partial \xi} \tag{191}$$

and the η-momentum equation becoming

$$\rho\frac{\partial u_\eta}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial u_\eta}{\partial \xi}+u_\eta\frac{\partial u_\eta}{\partial \eta}-K_{\eta\xi}u_\xi^2\right]=\rho f_{body,\eta}-\left(\frac{1}{h_\eta}\right)\frac{\partial p}{\partial \eta} \tag{192}$$

Energy The energy equation uses (110) and the curvilinear vector operations, notice that the stress tensor energy dissipation as well as the heat production and conduction terms disappear due to the assumptions of inviscid and adiabatic, to become

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+\left(\frac{1}{h_\eta}\right)u_\eta\frac{\partial H}{\partial \eta}\right] \tag{193}$$
$$=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}\right]$$

Applying the geodesic coordinate system conditions, this becomes

$$\rho\frac{\partial H}{\partial t}+\rho\left[\left(\frac{1}{h_\xi}\right)u_\xi\frac{\partial H}{\partial \xi}+u_\eta\frac{\partial H}{\partial \eta}\right]=\frac{\partial p}{\partial t}+\frac{\partial Q}{\partial t}+\rho\left[u_\xi f_{body,\xi}+u_\eta f_{body,\eta}\right] \qquad (194)$$

APPENDIX B

THREE POINT ARC FORMULATION

This appendix develops a closed form solution for the equation described by three points in $\mathscr{R}^2$.

Given three non-collinear points, $\{\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c\}$, in $\mathscr{R}^2$ then they form a circle of radius R with the center of the circle at $\boldsymbol{x}_0$. Thus, each point solves the following equation

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \tag{195}$$

Substituting the three points into (195) and multiplying out the squared terms yields

$$\begin{bmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{bmatrix} \begin{bmatrix} 2x_0 \\ 2y_0 \\ R^2 - x_0^2 - y_0^2 \end{bmatrix} = \begin{bmatrix} x_a^2 + y_a^2 \\ x_b^2 + y_b^2 \\ x_c^2 + y_c^2 \end{bmatrix} \tag{196}$$

Since (196) is a simple linear algebra equation, Kramer's rule (see for example [10] for

more information on Kramer's rule) can be used to solve (196) and get

$$x_0 = \frac{\begin{vmatrix} x_a^2+y_a^2 & y_a & 1 \\ x_b^2+y_b^2 & y_b & 1 \\ x_c^2+y_c^2 & y_c & 1 \end{vmatrix}}{2\begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}} \tag{197a}$$

$$y_0 = \frac{\begin{vmatrix} x_a & x_a^2+y_a^2 & 1 \\ x_b & x_b^2+y_b^2 & 1 \\ x_c & x_c^2+y_c^2 & 1 \end{vmatrix}}{2\begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}} \tag{197b}$$

$$R^2 - x_0^2 - y_0^2 = \frac{\begin{vmatrix} x_a & y_a & x_a^2+y_a^2 \\ x_b & y_b & x_b^2+y_b^2 \\ x_c & y_c & x_c^2+y_c^2 \end{vmatrix}}{\begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}} \tag{197c}$$

The location of the center of the circle is given by (197a) and (197b). In order to find the

radius of the circle, (197a) and (197b) are substituted into (197c) to get

$$R^2 = \frac{\begin{vmatrix} x_a^2+y_a^2 & y_a & 1 \\ x_b^2+y_b^2 & y_b & 1 \\ x_c^2+y_c^2 & y_c & 1 \end{vmatrix}^2 + \begin{vmatrix} x_a & x_a^2+y_a^2 & 1 \\ x_b & x_b^2+y_b^2 & 1 \\ x_c & x_c^2+y_c^2 & 1 \end{vmatrix}^2 + 4 \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix} \begin{vmatrix} x_a & y_a & x_a^2+y_a^2 \\ x_b & y_b & x_b^2+y_b^2 \\ x_c & y_c & x_c^2+y_c^2 \end{vmatrix}}{4 \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}^2} \tag{198}$$

When (198) is multiplied out and simplified, it becomes

$$R^2 = \frac{\left[(x_a-x_b)^2+(y_a-y_b)^2\right]\left[(x_a-x_c)^2+(y_a-y_c)^2\right]\left[(x_c-x_b)^2+(y_c-y_b)^2\right]}{4\left[x_c(y_a-y_b)+x_b(y_c-y_a)+x_a(y_b-y_c)\right]^2} \tag{199}$$

R can now be found by taking the square root of (199) to get

$$R = \pm\frac{\sqrt{\left[(x_a-x_b)^2+(y_a-y_b)^2\right]\left[(x_a-x_c)^2+(y_a-y_c)^2\right]\left[(x_c-x_b)^2+(y_c-y_b)^2\right]}}{2\left[x_c(y_a-y_b)+x_b(y_c-y_a)+x_a(y_b-y_c)\right]} \tag{200}$$

Finding x_0 and y_0 requires expanding the determinates in (197a) and (197b) to get

$$x_0 = \frac{(x_c^2+y_c^2)(y_a-y_b)+(x_b^2+y_b^2)(y_c-y_a)+(x_a^2+y_a^2)(y_b-y_c)}{2\left[x_c(y_a-y_b)+x_b(y_c-y_a)+x_a(y_b-y_c)\right]} \tag{201a}$$

$$y_0 = -\frac{(x_c^2+y_c^2)(x_a-x_b)+(x_b^2+y_b^2)(x_c-x_a)+(x_a^2+y_a^2)(x_b-x_c)}{2\left[x_c(y_a-y_b)+x_b(y_c-y_a)+x_a(y_b-y_c)\right]}$$

APPENDIX C

NACA 4-DIGIT AIRFOIL CURVATURE

This appendix develops the curvature equation for the NACA 4-digit airfoil for both the cambered and symmetric airfoils. First, the equations describing the airfoil surface is presented. This is followed by the development of the equations required in the curvature calculation for the general cambered airfoil. Finally, the relatively simpler curvature equation is developed for the non-cambered (i.e. symmetric) 4-digit airfoil.

Airfoil Description

The standard equation for the NACA 4-digit-series airfoil is represented by a four-digit number, $qnxx$, where q and n represent the camber specification and xx represents the thickness-chord ratio, $t_c = xx/100$. The standard equation for the airfoil can be obtained from several references such as [1] and is defined by starting with symmetric airfoil representation

$$y_s = \frac{t_c}{0.20}\left(a_0\sqrt{\bar{x}} + a_1\bar{x} + a_2\bar{x}^2 + a_3\bar{x}^3 + a_4\bar{x}^4\right) \tag{202}$$

and the description of the camber line as

$$
\begin{aligned}
p &= \frac{q}{100} \\
q &= \frac{n}{10} \\
y_c &= \begin{cases} \frac{m}{p^2}\left(2px - x^2\right) & \text{if } x < p; \\ \frac{m}{(1-p)^2}\left[(1-2p) + 2px - x^2\right] & \text{if } x \geq p. \end{cases}
\end{aligned}
\tag{203}
$$

Next, the airfoil surface coordinates can be represented by a combination of the symmetric airfoil equation (202) and the camber line equation (203) as the following set of equations

$$
\begin{aligned}
\tan\theta &= \frac{y_c}{x} \\
\bar{x} &= \begin{cases} x - y_s \sin\theta, & \text{upper surface;} \\ x + y_s \sin\theta, & \text{lower surface.} \end{cases} \\
\bar{y} &= \begin{cases} y_c + y_s \cos\theta, & \text{upper surface;} \\ y_c - y_s \cos\theta, & \text{lower surface.} \end{cases}
\end{aligned}
\tag{204}
$$

where $\bar{x}$ and $\bar{y}$ are the non-dimensionalized airfoil coordinates. The a coefficients in the symmetric airfoil equation are defined by the following boundary conditions for a thickness ratio 0.20 symmetric airfoil, NACA-0020:

1. Maximum Ordinate – The maximum ordinate occurs at $\bar{x} = 0.30$ and is $\bar{y} = 0.10$

2. Trailing Edge Ordinate – The trailing edge ordinate is $\bar{y} = 0.002$ at $\bar{x} = 1.0$

3. Trailing Edge Slope – The trailing edge slope is $|d\bar{y}/d\bar{x}| = 0.234$

4. Nose Shape – The shape of the nose is defined as $\bar{x} = 0.1$ and $\bar{y} = 0.078$

Applying these constraints, the coefficients are

$$a_0 = 0.2969 \qquad a_1 = -0.1260 \qquad a_2 = -0.3516 \qquad (205)$$
$$a_3 = 0.2843 \qquad a_4 = -0.1015$$

Cambered Airfoil Curvature

To start developing the cambered airfoil curvature equation, the standard definition of the radius of curvature, see [74], is presented here

$$K(\xi)^{-1} = R(\xi) = \frac{\left[1 + \left(\frac{d\bar{y}}{d\bar{x}}\Big|_{\bar{x}=\xi}\right)^2\right]^{3/2}}{\frac{d^2\bar{y}}{d\bar{x}^2}\Big|_{\bar{x}=\xi}} \qquad (206)$$

where K is the curvature and R is the radius of curvature. To find the first and second derivatives of the airfoil curve that are required in the curvature equation, it is convenient to use the chain rule to obtain the following relations

$$\frac{d\bar{y}}{d\bar{x}} = \frac{d\bar{y}/dx}{d\bar{x}/dx} \qquad (207)$$
$$\frac{d^2\bar{y}}{d\bar{x}^2} = \frac{\left(d^2\bar{y}/dx^2\right)\left(d\bar{x}/dx\right) - \left(d^2\bar{x}/dx^2\right)\left(d\bar{y}/dx\right)}{\left(d\bar{x}/dx\right)^3}$$

Combining equations (207) and (206) yields the following

$$K(\xi)^{-1} = \frac{\left[\left(d\bar{x}/dx\right)^2 + \left(d\bar{y}/dx\right)^2\right]^{3/2}}{\left(d^2\bar{y}/dx^2\right)\left(d\bar{x}/dx\right) - \left(d^2\bar{x}/dx^2\right)\left(d\bar{y}/dx\right)} \qquad (208)$$

Now the curvature equation is in terms of the independent coordinate x. Using equations (204) to develop the derivatives needed for equation (208) yields

$$\begin{aligned}
\frac{d\bar{x}}{dx} &= 1 \mp \left[\frac{dy_s}{dx}\sin\theta + y_s\frac{d\theta}{dx}\cos\theta\right] \\
\frac{d^2\bar{x}}{dx^2} &= \mp\left[\frac{d^2y_s}{dx^2}\sin\theta + 2\frac{dy_s}{dx}\frac{d\theta}{dx}\cos\theta + y_s\frac{d^2\theta}{dx^2}\cos\theta - y_s\left(\frac{d\theta}{dx}\right)^2\sin\theta\right] \\
\frac{d\bar{y}}{dx} &= \frac{dy_c}{dx} \pm \left[\frac{dy_s}{dx}\cos\theta - y_s\frac{d\theta}{dx}\sin\theta\right] \\
\frac{d^2\bar{y}}{dx^2} &= \frac{d^2y_c}{dx^2} \pm \left[\frac{d^2y_s}{dx^2}\cos\theta - 2\frac{dy_s}{dx}\frac{d\theta}{dx}\sin\theta - y_s\frac{d^2\theta}{dx^2}\sin\theta - y_s\left(\frac{d\theta}{dx}\right)^2\cos\theta\right]
\end{aligned} \tag{209}$$

where the upper sign in $\pm$ and $\mp$ refers to the upper surface and the lower sign refers to the lower surface. Applying these to the numerator and denominator of the curvature equation (208) yields

$$K(\xi)^{-1} = \frac{N}{D}$$

where
$$\begin{aligned}
N = 1 &+ \left(\frac{dy_c}{dx}\right)^2 + \left(\frac{dy_s}{dx}\right)^2 + y_s^2\left(\frac{d\theta}{dx}\right)^2 \\
&\pm 2\left[\frac{dy_s}{dx}\left(\frac{dy_c}{dx}\cos\theta - \sin\theta\right) - y_s\frac{d\theta}{dx}\left(\frac{dy_c}{dx}\sin\theta + \cos\theta\right)\right]
\end{aligned} \tag{210}$$

and
$$\begin{aligned}
D = &\frac{d^2y_c}{dx^2} + 2\left(\frac{dy_s}{dx}\right)^2\frac{d\theta}{dx} + y_s\frac{dy_s}{dx}\frac{d^2\theta}{dx^2} - y_s\frac{d^2y_s}{dx^2}\frac{d\theta}{dx} + y_s^2\left(\frac{d\theta}{dx}\right)^3 \\
&\pm\left\{\left[\frac{d^2y_s}{dx^2} - y_s\left(\frac{d\theta}{dx}\right)^2\right]\left(\cos\theta + \frac{dy_c}{dx}\sin\theta\right)\right. \\
&\quad + \left(2\frac{dy_s}{dx}\frac{d\theta}{dx} + y_s\frac{d^2\theta}{dx^2}\right)\left(\frac{dy_c}{dx}\cos\theta - \sin\theta\right) \\
&\quad \left. - \frac{d^2y_c}{dx^2}\left(\frac{dy_s}{dx}\sin\theta + y_s\frac{d\theta}{dx}\cos\theta\right)\right\}
\end{aligned}$$

Using the definition of θ, the following derivatives can be found

$$\frac{d\theta}{dx} = \frac{\cos^2\theta \frac{dy_c}{dx} - \sin\theta\cos\theta}{x} \tag{211}$$
$$\frac{d^2\theta}{dx^2} = \frac{\cos^2\theta\left(\frac{d^2y_c}{dx^2}x - \frac{dy_c}{dx}\right) + \cos\theta\sin\theta\left(1 - 2\frac{d\theta}{dx}\frac{dy_c}{dx}x\right) + \left(\sin^2\theta - \cos^2\theta\right)\frac{d\theta}{dx}x}{x^2}$$

Finally, the first and second derivatives of the symmetric airfoil equation is

$$\frac{dy_s}{dx} = \frac{t_c}{0.20}\left(\frac{a_0}{\sqrt{\bar{x}}} + a_1 + 2a_2\bar{x} + 3a_3\bar{x}^2 + 4a_4\bar{x}^3\right) \tag{212}$$
$$\frac{d^2y_s}{dx^2} = \frac{t_c}{0.20}\left(\frac{a_0}{\bar{x}^{3/2}} + 2a_2 + 6a_3\bar{x} + 12a_4\bar{x}^2\right)$$

and for the camber line equation

$$\frac{dy_c}{dx} = \begin{cases} \frac{2m}{p^2}(p-x) & \text{if } x < p; \\ \frac{2m}{(1-p)^2}(p-x) & \text{if } x \geq p. \end{cases} \tag{213}$$

$$\frac{d^2y_c}{dx^2} = \begin{cases} -\frac{2m}{p^2} & \text{if } x < p; \\ -\frac{2m}{(1-p)^2} & \text{if } x \geq p. \end{cases}$$

Combining equations (202), (203), (211), (212) and (213) with equation (210) yields the curvature equation for the NACA 4-digit airfoil.

Symmetric Airfoil Curvature

Developing the symmetric airfoil curvature starts with simplifying the general curvature equations developed above for the case where $y_c = 0$. The following is equations (202), (203),

(211), (212) and (213) with the symmetric limitation applied

$$
\begin{aligned}
y_s &= \frac{t_c}{0.20}\left(a_0\sqrt{\bar{x}} + a_1\bar{x} + a_2\bar{x}^2 + a_3\bar{x}^3 + a_4\bar{x}^4\right) \\
\frac{dy_s}{dx} &= \frac{t_c}{0.20}\left(\frac{a_0}{\sqrt{\bar{x}}} + a_1 + 2a_2\bar{x} + 3a_3\bar{x}^2 + 4a_4\bar{x}^3\right) \\
\frac{d^2y_s}{dx^2} &= \frac{t_c}{0.20}\left(\frac{a_0}{\bar{x}^{3/2}} + 2a_2 + 6a_3\bar{x} + 12a_4\bar{x}^2\right) \\
y_c &= \frac{dy_c}{dx} = \frac{d^2y_c}{dx^2} = 0 \\
\theta &= \frac{d\theta}{dx} = \frac{d^2\theta}{dx^2} = 0
\end{aligned}
\tag{214}
$$

Also, notice that $\bar{x}$ and $\bar{y}$ simply become x and y_s, respectively. Applying these simplified equations to the curvature equation (210) yields

$$
K(\xi)^{-1} = \frac{\left[1 + \left(\frac{dy_s}{dx}\right)^2\right]^{3/2}}{\pm\frac{d^2y_s}{dx^2}} \tag{215}
$$

which is just the curvature equation for the symmetric airfoil with the $\pm$ signifying the upper or lower surface. This can be simplified further by substituting the equations for the y_s terms to get

$$
K(x)^{-1} = \frac{\left(\frac{t_c}{0.20}\right)^2\left[4\left(\frac{0.20}{t_c}\right)^2\bar{x} + \left(a_0 + 2a_1\sqrt{x} + 4a_2x^{3/2} + 6a_3x^{5/2} + 8a_4x^{7/2}\right)^2\right]^{3/2}}{2\left(-a_0 + 8a_2x^{3/2} + 24a_3x^{5/2} + 48a_4x^{7/2}\right)} \tag{216}
$$

APPENDIX D

NUMERICAL CONSERVATION

This appendix demonstrates the conservation properties of the numerical scheme with and without the solid surface treatment.

Since the original NASCART-GT solver is based on a finite volume scheme solving the Euler and Navier-Stokes equations in conservation form, it is a conservative scheme (see Chapter II for details). The solid boundary treatment discussed in Chapter III removes the surface cells from the finite volume scheme, and there is no assurance that the surface cell treatment remains conservative. Therefore, the use of the solid boundary treatment makes the overall scheme non-conservative.

In order to address how much impact the non-conservative solid boundary treatment has on the overall conservation of the scheme, the incompressible, inviscid cylinder case discussed on page 105 was used to determine this impact. To determine the degree to which this scheme is non-conservative, a control volume is place around the entire computational domain, and the net flux through the control volume is calculated. Figure 90 shows a schematic of the control used to calculate the net fluxes of the conserved quantities.

A complete finite volume solution to this case, i.e. using the finite volume formulation for the surface cells instead of the solid boundary treatment, is used establish the numerical

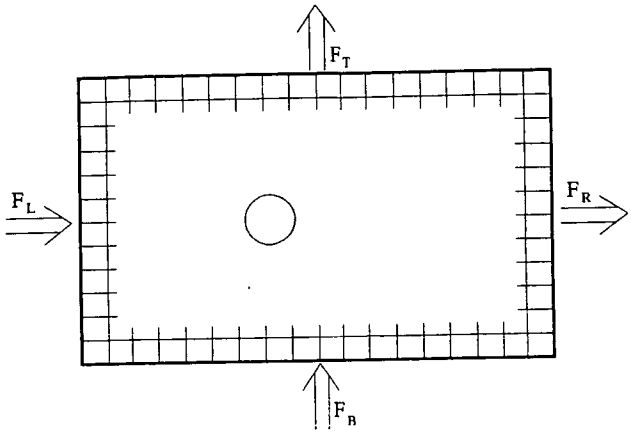


Figure 90: Incompressible Cylinder Control Volume

conservation properties of this scheme. While the net flux should be zero for this case, numerical errors will cause it to be non-zero. Table 15 shows the results for this. The mass net flux is about 0.6% of the flux into the control volume and the energy net flux is about 0.3%. The same net flux calculation for the curved wall boundary condition solution is also shown in table 15. The mass net flux for this case is again about 0.6% and the energy net flux is about 0.3%. Also, there is virtually no difference between the 0.2% relative difference between the net mass fluxes and 0.2% relative difference for the net energy flux.

Table 15: Net Fluxes for Incompressible Cylinder

		Finite-Volume	Curved Wall
	F_{in}	F_{net}	F_{net}
mass	0.132954	7.87867E-04	7.85989E-04
energy	47.6208	0.148208	0.147871

While the solid boundary treatment makes this scheme formally non-conservative, the net flux differences between the conservative finite volume scheme and the solid boundary treatment are negligible.

Bibliography

[1] I. H. Abbott and A. E. von Doenhoff. *Theory of Wing Sections*. Dover Publications, Inc., New York, 1959.

[2] M. J. Aftosmis. Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries. In *Lecture Notes for 28th Computational Fluid Dynamics Lecture Series*. von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, March 1997.

[3] M. J. Aftosmis, M. J. Berger, and G. Adomavicius. A Parallel Cartesian Approach for External Aerodynamics of Vehicles with Complex Geometry. In *Thermal and Fluids Analysis Workshop*, September 1999.

[4] M. J. Aftosmis, M. J. Berger, and G. Adomavicius. A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries. In *38th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2000. AIAA. AIAA-2000-0808.

[5] M. J. Aftosmis, D. Gaitonde, and T. S. Tavares. Behavior of Linear Reconstruction Techniques on Unstructured Meshes. *AIAA Journal*, 33(11):2038–2049, November 1995.

[6] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 142(1):1–46, 1998.

[7] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Pub. Corp., McGraw-Hill, New York, 2nd edition, 1984.

[8] J. D. Anderson, Jr. *Modern Compressible Flow with Historical Perspective*. McGraw-Hill, Inc., New York, 2nd edition, 1990.

[9] W. K. Anderson, J. L. Thomas, and B. van Leer. A Comparison of Finite Volume Flux Vector Splitting for the Euler Equations. In *AIAA 23st Aerospace Sciences Meeting*, Reno, NV, January 1985. AIAA-85-0122.

[10] H. Anton and C. Rorres. *Elementary Linear Algebra, Applications Version*. John Wiley & Sons, Inc., New York, 6th edition, 1991.

[11] P. Arminjon and A. Madrane. Staggered Mixed Finite Volume/Finite Element Method for the Navier-Stokes Equations. *AIAA Journal*, 37(12):1558–1571, December 1999.

[12] N. Ashgriz and J.Y. Poo. FLAIR: Flux Line-segment Model for Advection and Interface Reconstruction. *Journal of Computational Physics*, 92(2):449–468, 1991.

[13] E. Atta. Component-Adaptive Grid Interfacing. In *19th Aerospace Sciences Meeting*, St. Louis, MO, January 1981. AIAA-81-0382.

[14] E. H. Atta and J. Vadyak. A Grid Interfacing Zonal Algorithm for Three Dimensional Transonic Flows About Aircraft Configurations. In *AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference*, St. Louis, MO, June 1982. AIAA-82-1017.

[15] E. H. Atta and J. Vadyak. Numerical Simulation of the Transonic Flowfield for Wing/Nacelle Configurations. In *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting*, San Diego, CA, October 1984. AIAA-84-2430.

[16] B. S. Baldwin and H. Lomax. Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows. In *AIAA 16th Aerospace Science Meeting*, Huntsville, AL, January 1978. AIAA-78-257.

[17] T. J. Barth and S. W. Linton. An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation. In *33rd Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 1995. AIAA. AIAA-95-0221.

[18] J. T. Batina. A Gridless Euler/Navier-Stokes Solution Algorithm for Complex-Aircraft Applications. In *AIAA 31th Aerospace Sciences Meeting*, Reno, NV, January 1993. AIAA-93-0333.

[19] S. A. Bayyuk, K. G. Powell, and B. van Leer. A Simulation Technique for 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA-93-3391-CP.

[20] T. Belytschko, Y. Y. Lu, and L. Gu. Element-Free Galerkin Methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, January 1994.

[21] J. A. Benek, P. G. Buning, and J. L. Steger. A 3-D Chimera Grid Embedding Technique. In *7th Computational Fluid Dynamics Conference*, Cincinnati, OH, July 1985. AIAA. AIAA-85-1523.

[22] M. J. Berger, M. J. Aftosmis, and G. Adomavicius. Parallel Multigrid on Cartesian Meshes with Complex Geometry. In *8th International Conference on Parallel CFD*, Trondheim, Norway, 2000.

[23] M. J. Berger and R. J. LeVeque. An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries. In *9th AIAA Computational Fluid Dynamics Conference*, Buffalo, NY, June 1989. AIAA-89-1930-CP.

[24] M. J. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53(1):484–512, 1984.

[25] J. J. Berten and M. L. Smith. *Aerodynamics for Engineers*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.

[26] W. L. Briggs. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[27] M. O. Bristeau, R. Glowinski, J. Periaux, and H. Viviand. Presentation of Problems and Discussion of Results. In M. O. Bristeau, R. Glowinski, J. Periaux, and H. Viviand, editors, *Numerical Simulations of Compressible Navier-Stokes Flows*, Notes on Numerical Fluid Mechanics, pages 1–40. Friedr. Vieweg & Sohn, 1987.

[28] L. Cambier. Computation of Viscous Transonic Flows Using an Unsteady Type Method and a Zonal Grid Refinement Technique. In M. O. Bristeau, R. Glowinski, J. Periaux, and H. Viviand, editors, *Numerical Simulations of Compressible Navier-Stokes Flows*, Notes on Numerical Fluid Mechanics, pages 105–122. Friedr. Vieweg & Sohn, 1987.

[29] J. E. Carter. Numerical Solutions of the Navier-Stokes Equations for the Supersonic Laminar Flow Over a Two-Dimensional Compression Corner. NASA Technical Report NASA-TR-R-385, NASA Langley Research Center, Hampton, VA, July 1972.

[30] J. E. Carter. A New Boundary-Layer Interaction Technique for Separated Flows. NASA TM-78690, June 1978.

[31] J. E. Carter. A New Boundary-Layer Inviscid Interaction Technique for Separated Flow. In *4th AIAA Computational Fluid Dynamics Conference*, Williamsburg, VA, July 1979. AIAA-79-1450.

[32] F. Casalini and A. Dadone. Computations of Viscous Flows Using a Multigrid Finite Volume Lamda Formulation. *Engineering Computations*, 16(7):767–786, 1999.

[33] T. Cebeci, R. W. Clark, K. C. Chang, N. D. Halsey, and K. Lee. Airfoils with Separation and the Resulting Wakes. *Journal of Fluid Mechanics*, 163:323–347, February 1986.

[34] L. T. Chen and M. N. Bui. An Interactive Scheme for Transonic Wing/Body Flows Based on Euler and Inverse Boundary-Layer Equations. In *AIAA 21st Fluid Dynamics, Plasma Dynamics and Lasers Conference*, Seattle, WA, June 1990. AIAA-90-1586.

[35] Y.-L. Chiang, B. van Leer, and K. G. Powell. Simulation of Unsteady Inviscid Flow on an Adaptively Refined Cartesian Grid. In *30th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 1992. AIAA. AIAA-92-0443.

[36] D. K. Clarke, M. D. Salas, and H. A. Hassan. Euler Calculations for Multielement Airfoils Using Cartesian Grids. *AIAA Journal*, 24(3):353–358, March 1986.

[37] E. Cohen. Some Mathematical Tools for a Modeler's Workbench. *IEEE Computer Graphics and Applications*, 3(7):63–66, October 1983.

[38] W. J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, University of Michigan, Ann Arbor, MI, 1993.

[39] W. J. Coirier. An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations. NASA Technical Memorandum 106754, NASA Lewis Research Center, Cleveland, OH, October 1994.

[40] W. J. Coirier and K. G. Powell. A Cartesian, Cell-Based Approach for Adaptively-Refined Solutions of the Euler and Navier-Stokes Equations. In *33rd Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 1995. AIAA. AIAA-95-0556.

[41] W. J. Coirier and K. G. Powell. Solution-Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows. *AIAA Journal*, 34(5):938–945, May 1996.

[42] P. Colella, R. Ferguson, and H. Glaz. Multifluid Algorithms for Eulerian Finite Difference Methods. Preprint, 1996.

[43] Cray Research, Inc., Eagan, MN. *CRAY T3D System Architecture Overview*, March 1994. HR-04033.

[44] A. Dadone and B. Grossman. Surface Boundary Conditions for the Numerical Solution of the Euler Equations. *AIAA Journal*, 32(2):285–293, February 1994.

[45] D. De Zeeuw and K. G. Powell. An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations. In *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, HI, June 1991. AIAA-91-1542-CP.

[46] D. L. De Zeeuw. *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm For Solution Of The Euler Equations*. PhD thesis, University of Michigan, Ann Arbor, MI, 1993.

[47] N. H. Decker, V. K. Naik, and M. Nicoules. Parallelization of Implicit Finite Difference Schemes in Computational Fluid Dynamics. ICASE Report 90-53, ICASE, Hampton, VA, August 1990. NASA/CR-182081.

[48] F. Deister and E. H. Hirschel. Self-Organizing Hybrid Cartesian Grid/Solution System with Multigrid. In *40th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2002. AIAA. AIAA-2002-0112.

[49] M. Delanaye, M. J. Aftosmis, M. J. Berger, Y. Liu, and T. H. Pulliam. Automatic Hybrid-Cartesian Grid Generation for High-Reynolds Number Flows around Complex Geometries. In *AIAA 37th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 1999. AIAA-99-0777.

[50] H. G. Dietz and T. I. Mattox. KLAT2's Flat Neighborhood Network. In *4th Annual Linux Showcase & Conference*, pages 91–100, Atlanta, GA, October 2000.

[51] W. Dietz, M. Fan, J. Steinhoff, and Y. Wenren. Application of Vorticity Confinement to the Predicition of the Flow Over Complex Bodies. In *AIAA 15th CFD Conference*, Anaheim, CA, June 2001. AIAA. AIAA-2001-2642.

[52] N. D. Domel and S. L. Karman, Jr. Splitfow: Progress in 3D CFD with Cartesian Omni-tree Grids for Complex Geometries. In *AIAA 38th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 2000. AIAA-2000-1006.

[53] M. Drela and M. B. Giles. Viscous-Inviscid Analysis of Transonic and Low Reynolds Number Airfoils. *AIAA Journal*, 25(10):1347–1354, October 1987.

[54] T. M. Eidson and G. Erlebacher. Implementation of a Fully-Balanced Periodic Tridiagonal Solver on a Parallel Distributed Memory Architecture. ICASE Report 94-37, ICASE, Hampton, VA, May 1994. NASA/CR-194919.

[55] B. Epstein, A. L. Luntz, and A. Nachshon. Multigrid Transonic Computations About Arbitrary Aircraft Configurations. *Journal of Aircraft*, 26(8):751–759, August 1989.

[56] B. Epstein, A. L. Luntz, and A. Nachshon. Cartesian Euler Method for Arbitrary Aircraft Configurations. *AIAA Journal*, 30(3):679–687, March 1992.

[57] E. A Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex. *Journal of Computational Physics*, 161(1):35–60, 2000.

[58] M. Fan, W. Dietz, Y. Wenren, and J. Steinhoff. Computing Complex Flows on Coarse Grids Using Vorticity Confinement. In *40th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2002. AIAA. AIAA-2002-0135.

[59] P. D. Frymier, Jr., H. A. Hassan, and M. D. Salas. Navier-Stokes Calculations Using Cartesian Grids: I. Laminar Flows. *AIAA Journal*, 26(10):1181–1188, October 1988.

[60] R. L. Gaffney, H. A. Hassan, and M. D. Salas. Euler Calculations for Wings Using Cartesian Grids. In *AIAA 25th Aerospace Sciences Meeting*, Reno, NV, January 1987. AIAA-87-0356.

[61] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Labs, Oak Ridge, TN, September 1994. ORNL/TM-12187.

[62] S. K. Godunov. A Finite Difference Method for the Computation of Discontinuous Solutions of the Equations of Fluid Dynamics. *Matematicheskii sbornik*, 47:357–393, 1959.

[63] S. K. Godunov. Reminiscences about Difference Schemes. *Journal of Computational Physics*, 153(1):6–25, 1999.

[64] D. Goldstein, R. Handler, and L. Sirovich. Modeling a No-Slip Flow Boundary with an External Force Field. *Journal of Computational Physics*, 105(2):354–366, 1993.

[65] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, September 1996.

[66] D. Heller. A Survey of Parallel Algorithms in Numerical Linear Algebra. *SIAM Review*, 20(4):740–777, October 1978.

[67] L. F. Henderson, P. Colella, and E. G. Pucket. On the Refraction of Shock Waves at a Slow-Fast Gas Interface. *Journal of Fluid Mechanics*, 224:1–27, March 1991.

[68] G. Hipper and D. Tavangarian. Advanced Workstation Cluster Architectures for Parallel Computing. *Journal of System Architecture*, 44:207–226, 1998.

[69] E. H. Hirschel and W. Kordulla. *Shear Flow in Surface-Oriented Coordinate*. Friedr. Vieweg & Sohn, Braunschweig, Germany, 1981.

[70] C. W. Hirt and B. D. Nichols. Volume of Fluid (VOF) Method for Dynamics of Free Boundaries. *Journal of Computational Physics*, 39(1):201–221, 1981.

[71] D. G. Holmes and S. D. Connell. Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids. In *9th AIAA Computational Fluid Dyamics Conference*, Buffalo, NY, June 1989. AIAA-89-1932-CP.

[72] L. Howarth. The Boundary Layer in Three Dimensional Flow. – Part I. Derivation of the Equations for Flow along a General Curved Surface. *Philosophical Magazine*, 42:239–243, March 1951. No. 326.

[73] G. Hu, B. Grossman, and J. Steinhoff. A Numerical Method for Vortex Confinement in Compressible Flow. In *AIAA 38th Aerospace Sciences Meeting*, Reno, NV, January 2000. AIAA-2000-0281.

[74] J. F. Hurley. *Calculus*. Wadsworth Publishing, Belmont, CA, 1987.

[75] A. Jameson. Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method. *Applied Mathematics and Computation*, 13(3–4):327–356, November 1983.

[76] D. C. Jespersen. Parallelism and OVERFLOW. NAS Technical Report NAS-98-013, NASA Ames Research Center, Moffett Field, CA, October 1998.

[77] Y. Kallinderis and S. Ward. Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations. In *10th Applied Aerodynamics Conference*, Palo Alto, CA, June 1992. AIAA-92-2721-CP.

[78] K.-H. Kao, M.-S. Liou, and C.-Y. Chow. Grid Adaption Using Chimera Composite Overlapping Meshes. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA-93-3389-CP.

[79] S. L. Karman, Jr. SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries. In *33rd Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 1995. AIAA. AIAA-95-0343.

[80] D. S. Katz, T. Cwik, B. H. Kwan, J. Z. Lou, P. L. Springer, T. L. Sterling, and P. Wang. An Assessment of a Beowulf System for a Wide Class of Analysis and Design Software. *Advances in Engineering Software*, 29(3–6):451–561, 1998.

[81] K. Kaups and T. Cebeci. Compressible Laminar Boundary Layers with Suction on Swept and Tapered Wings. *Journal of Aircraft*, 14(7):661–667, July 1977.

[82] J. KIm, K. Kim, and H. Choi. An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries. *Journal of Computational Physics*, 171(1):132–150, 2001.

[83] M. Kremenetsky, T. Tysinger, and S. Posey. Considerations for Parallel CFD Enhancements on SGI ccNUMA and Cluster Architectures. In *10th Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, April 2001.

[84] K. A. Kurbatskii and C. K. W. Tam. Cartesian Boundary Treatment of Curved Walls for High-Order Computational Aeroacoustics Schemes. *AIAA Journal*, 35(1):133–140, January 1997.

[85] C. L. Ladson, C. W. Brooks, Jr., A. S. Hill, and D. W. Sproles. Computer Program To Obtain Ordinates for NACA Airfoils. NASA Technical Memorandum 4741, NASA Langley Research Center, Hampton, VA, December 1996.

[86] P. R. Lahur and Y. Nakamura. Simulation of Flow Around Moving 3D Body on Unstructured Cartesian Body. In *15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, June 2001. AIAA-2001-2605.

[87] M.-C. Lai and C. S. Peskin. An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity. *Journal of Computational Physics*, 160(12):705–719, 2000.

[88] M. Lesionne and C. Farhat. Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable Meshes, and Their Impact on Aeroelastic Computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[89] R. J. LeVeque. A Large Time Step Generalization of Godunov's Method for Systems of Conservation Laws. *SIAM Journal on Numerical Analysis*, 22(5):1051–1073, December 1985.

[90] L. Lijewski and N. Suhs. Chimera-Eagle Store Separation. In *AIAA Atmospheric Flight Mechanics Conference*, Hilton Head, SC, August 1992. AIAA-94-1925-CP.

[91] J.-L. Liu and S.-J. Su. A Potentially Gridless Solution Method for the Compressible Euler/Navier-Stokes Equations. In *34th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 1996. AIAA. AIAA-96-0526.

[92] R. Löhner. Some Useful Renumbering Stategies for Unstructured Grids. *International Journal for Numerical Methods in Engineering*, 36(19):3259–3270, October 1993.

[93] R. Löhner and M. Galle. Minimization of Indirect Addressing for Edge-Based Field Solvers. In *40th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2002. AIAA. AIAA-2002-0967.

[94] G. R. Luecke, M. Kraeva, and L. Ju. Comparing the Performance of MPICH with Cray's MPI and with SGI's MPI. *Concurrency and Computation: Practice and Experience*, 2002. Accepted April 10, 2002.

[95] S. Majumdar, G. Iaccarino, and P. Durbin. RANS Solvers with Adaptive Structured Boundary Non-Conforming Grids. Annual Research Briefs 208782, Center for Turbulence Research, Stanford University, Stanford, CA, 2001.

[96] D. J. Mavriplis. Large-Scale Parallel Viscous Flow Computations using an Unstructured Multigrid Algorithm. ICASE Report 99-44, ICASE, Hampton, VA, November 1999. NASA/CR-1999-209724.

[97] D. J. Mavriplis. Parallel Performance Investigations of an Unstructured Mesh Navier-Stokes Solver. ICASE Report 2000-13, ICASE, Hampton, VA, March 2000. NASA/CR-2000-210088.

[98] D. J. Mavriplis. Parallel Unstructured Mesh Analysis of High-Lift Configurations. In *AIAA 38th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 2000. AIAA-2000-0923.

[99] D. J. Mavriplis and S. Pirzadeh. Large-Scale Parallel Unstructured Mesh Computations for 3D High-Lift Analysis. ICASE Report 99-09, ICASE, Hampton, VA, February 1999. NASA/CR-1999-208999.

[100] R. Meakin. Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA-93-3350-CP.

[101] R. Meakin. On the Spatial and Temporal Accuracy of Overset Grid Methods for Moving Body Problems. In *12th Applied Aerodynamics Conference*, Colorado Springs, CO, June 1994. AIAA-94-1925-CP.

[102] R. L. Meakin. An Efficient Means of Adaptive Refinement Within Systems of Overset Grids. In *12th AIAA Computational Fluid Dynamics Conference*, San Diego, CA, June 1995. AIAA-95-1722-CP.

[103] R. L. Meakin. On Adaptive Refinement and Overset Structured Grids. In *13th AIAA Computational Fluid Dynamics Conference*, Snowmass Village, CO, June 1997. AIAA-97-1858-CP.

[104] J. E. Melton, M. J. Berger, M. J. Aftosmis, and M. D. Wong. 3D Applications of a Cartesian Grid Euler Method. In *33rd Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 1995. AIAA. AIAA-95-0853.

[105] J. E. Melton, F. Y. Enomoto, and M. J. Berger. 3D Automatic Cartesian Grid Generation for Euler Flows. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA-93-3386-CP.

[106] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard (Version 1.1)*, June 1995. http://www.mpi-forum.org.

[107] Message Passing Interface Forum. *MPI-2: Extension to the Message-Passing Interface*, July 1997. http://www.mpi-forum.org.

[108] G. H. Miller and E. G. Puckett. A High-order Godunov Method for Multiple Condensed Phases. *Journal of Computational Physics*, 128(1):134–164, 1996.

[109] R. A. Mitcheltree, M. D. Salas, and H. A. Hassan. Grid Embedding Technique Using Cartesian Grids for Euler Solutions. *AIAA Journal*, 26(6):754–756, June 1988.

[110] J. Mohd-Yosuf. Combined Immersed-Boundary/B-spline Methods for Simulations of Flow in Complex Geometries. Annual research briefs, Center for Turbulence Research, Stanford University, Stanford, CA, 1997.

[111] J. Mohd-Yosuf. Development of Immersed Boundary Methods for Complex Geometries. Annual research briefs, Center for Turbulence Research, Stanford University, Stanford, CA, 1998.

[112] M. Moultin and J. Steinhoff. A Technique for the Simulation of Stall with Coarse-Grid CFD Methods. In *AIAA 38th Aerospace Sciences Meeting*, Reno, NV, January 2000. AIAA-2000-0277.

[113] B. Müller, T. Berglind, and A. Rizzi. Implicit Central Difference Simulation of Compressible Navier-Stokes Flow Over a NACA0012 Airfoil. In M. O. Bristeau, R. Glowinski, J. Periaux, and H. Viviand, editors, *Numerical Simulations of Compressible Navier-Stokes Flows*, Notes on Numerical Fluid Mechanics, pages 183–200. Friedr. Vieweg & Sohn, 1987.

[114] M. Murayama and K. Nakahashi. Numerical Simulation of Vortical Flows Using Vorticity Confinement Coupled with Unstructured Grid. In *39th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2001. AIAA-2001-0606.

[115] S. M. Murman, M. J. Aftosmis, and M. J. Berger. Numerical Simulation of Rolling-Airframes Using a Multi-Level Cartesian Method. In *20th AIAA Applied Aerodynamics Conference*, St. Louis, MO, June 2002. AIAA. AIAA-2002-2798.

[116] NASA Ames Research Center. Cluster T27B Existing Configuration. Internal NASA Ames System Documentation, 29 November 2001.

[117] NASA Ames Research Center. NAS O2K Cluster Hardware Information. On-Line Documentation, 7 December 2001. http://www.nas.nasa.gov/Groups/SciCon/O2K/Hardware/index.html.

[118] NASA Ames Research Center. NAS O2K Cluster Software Information. On-Line Documentation, 7 December 2001. http://www.nas.nasa.gov/Groups/SciCon/O2K/Software/index.html.

[119] North Atlantic Treaty Organization. Test Cases for Inviscid Flow Field Methods. Technical Report AGARD-AR-211, North Atlantic Treaty Organization Advisory Group for Aerospace Research and Development, 1985. Report of Fluid Dynamics Panel Working Group 07.

[120] E. Oktay, N. Alemdaroglu, E. Tarhan, P. Champigny, and P. d'Espiney. Euler and Navier-Stokes Solutions for Missiles at High Angle of Attack. *Journal of Spacecraft and Rockets*, 36(6):850–858, November 1999.

[121] OpenMP Architecture Review Board. *OpenMP C and C++ Application Program Interface: Version 1.0*, October 1998. http://www.openmp.org.

[122] OpenMP Architecture Review Board. *OpenMP FORTRAN Application Program Interface: Version 2.0*, November 2000. http://www.openmp.org.

[123] P. S. Pacheco. *Parallel Computing with MPI.* Morgan Kaufmann Publishers, Inc., San Francisco, 1997.

[124] S. A. Pandya and M. J. Aftosmis. Computation of External Aerodynamics for a Canard Rotor/Wing Aircraft. In *39th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2001. AIAA-2001-0997.

[125] D. G. Pearce, S. A. Stanley, F. W. Martin, Jr., R. J. Gomez, G. J. Le Beau, P. G. Buning, W. M. Chan, I.-T. Chiu, A. Wulf, and V. Akdag. Development of a Large Scale Chimera Grid System for the Space Shuttle Launch Vehicle. In *31st Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 1993. AIAA. AIAA-93-0533.

[126] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions. http://citeseer.nj.nec.com/pember93adaptive.html, 1993.

[127] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 1993. AIAA-93-3385-CP.

[128] C. S. Peskin. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics*, 25:220–252, 1977.

[129] C. S. Peskin. The Fluid Dynamics of Heart Valves: Experimental, Theoretical, and Computational Methods. *Annual Review of Fluid Mechanics*, 14:235–259, 1982.

[130] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Journal on Matrix Analysis & Applications*, 11(3):430–452, July 1990.

[131] M. A. Potsdam. An Unstructured Mesh Euler and Interactive Boundary Layer Method for Complex Configurations. In *AIAA 12th Applied Aerodynamics Conference*, Colorado Springs, CO, June 1994. AIAA-94-1844.

[132] T. H. Pulliam. Euler and Thin layer Navier Stokes Codes: ARC2D, ARC3D. In K. C. Reddy and J. S. Steinhoff, editors, *Computational Fluid Dynamics, A workshop Held at The University of Tennessee Space Institute*, pages 15.1–15.85. University of Tennessee Space Institute, Tullahoma, TN, March 1984. UTSI Publication No E02-4005-023-84.

[133] T. H. Pulliam and J. T. Barton. Euler Computations of AGARD Working Group 07 Airfoil Test Cases. In *AIAA 23rd Aerospace Sciences Meeting*, Reno, NV, January 1985. AIAA-85-0018.

[134] T. H. Pulliam and J. L. Steger. On Implicit Finite-Difference Simulations of Three Dimensional Flow. In *AIAA 16th Aerospace Sciences Meeting*, Huntsville, AL, January 1978. AIAA-78-10.

[135] J. W. Purvis and J. E. Burkhalter. Prediction of Critical Mach Number for Store Configurations. *AIAA Journal*, 17(11):1170–1177, November 1979.

[136] J. J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics*. PhD thesis, Cranfield Institute of Technology, Wiltshire, UK, January 1991.

[137] J. J. Quirk. AMRITA - A Computational Facility (for CFD Modelling). In *Lecture Notes for 29th Computational Fluid Dynamics Lecture Series*. von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, February 1998.

[138] S. A. Ragab. Euler/Boundary Layer Solutions for Vortex Separation from Smooth Surfaces. In *AIAA 23rd Aerospace Sciences Meeting*, Reno, NV, January 1985. AIAA-85-0016.

[139] M. Rangarajan and L. Iftode. Software Distributed Shared Memory over Virtual Interface Architecture: Implementation and Performance. In *4th Annual Linux Showcase & Conference*, pages 341–352, Atlanta, GA, October 2000.

[140] A. Rizzi. Numerical Implementation of Solid-Body Boundary Conditions for the Euler Equations. *Zeitschrift für angewandte Mathematik und Mechanik*, 58:T301–T304, 1978.

[141] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[142] P. L. Roe and J. Pike. Efficient Construction and Utilisation of Approximate Riemann Solutions. In *Computing Methods in Applied Science and Engineering*. North–Holland, 1994.

[143] D. Roose and R. Van Driessche. Parallel Computers and Parallel Algorithms for CFD: An Introduction. In *Special Coarse on Parallel Computing in CFD*, Rhode-Saint-Genèse, Belgium and NASA Ames, Moffett Field, CA, October 1995. North Atlantic Treaty Organization. AGARD Report-807.

[144] J. K. Salmon, M. S. Warren, and G. S. Winkelmans. Fast Parallel Tree Codes for Gravitational and Fluid Dynamical N-Body Problems. *International Journal of Supercomputer Applications and High Performance Computing*, 8(2):129–142, 1994.

[145] R. Scardovelli and S. Zaleski. Direct Numerical Simulation of Free-Surface and Interfacial Flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.

[146] V. Schmitt and F. Charpin. Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers. In *Experimental Data Base for Computer Program Assessment*. North Atlantic Treaty Organization, May 1979. AGARD Advisory Report 138.

[147] M. S. Selig and J. J. Guglielmo. High-Lift Low Reynolds Number Airfoil Design. *Journal of Aircraft*, 34(1):72–79, January 1997.

[148] D. Sharov, H. Luo, J. D. Baum, and R. Löhner. Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-Based Parallel Computers. In *AIAA 38th Aerospace Sciences Meeting & Exhibit*, Reno, NV, January 2000. AIAA-2000-0927.

[149] S. T. Shaw and N. Qin. Unsteady Flow Around Helicopter Rotor Blade Sections in Forward Flight. *Aeronautical Journal*, 103:35–44, 1999.

[150] H. D. Simon. Partitioning of Unstructured Problems for Parallel Processing. In *Symposium on Parallel Methods on Large-Scale Structural Analysis and Physics Applications*, Hampton, VA, February 1991.

[151] M. Snir, P. Hochschild, D. D. Frye, and K. J. Gildea. The Communication Software and Parallel Environment of the IBM SP2. *IBM Systems Journal on Scalable Parallel Computing*, 34(2):205–221, 1995.

[152] J. L. Steger. Implicit Finite Difference Simulation of Flow About Arbitrary Geometries With Application to Airfoils. In *AIAA 10th Fluid & Plasma Dynamics Conference*, Albuquerque, NM, June 1977. AIAA-77-665.

[153] J. L. Steger, F. C. Dougherty, and J. A. Benek. A Chimera Grid Scheme. In K. N. Ghia and U. Ghia, editors, *Advances in Grid Generation: Presented at the Applied Mechanics, Bioengineering, and Fluids Engineering Conference*, volume 5, pages 59–69. The Fluid Engineering Division, ASME, Houston, TX, June 1983.

[154] J. L. Steger and W. R. Van Dalsem. Developments in the Simulation of Separated Flows Using Finite Difference Methods. In *AIAA 3rd Symposium on Numerical and Physical Aspects of Aerodynamic Flows*, pages 1–20, Long Beach, CA, January 1985.

[155] J. Steinhoff and D. Underhill. Modification of the Euler Equations for "Vorticity Confinement": Application to the Computation of Interacting Vortex Rings. *Physics of Fluids*, 6(8):2378–2744, August 1994.

[156] J. Steinhoff, W. Yonghu, and W. Lesong. Efficient Computation of Separating High Reynolds Number Incompressible Flows Using Vorticity Confinement. In *AIAA 37th Aerospace Sciences Meeting*, Reno, NV, January 1999. AIAA-99-3316.

[157] T. Sterling, D. Savarese, D. J. Becker, B. Fryxell, and K. Olson. Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation. In *Fourth IEEE International Symposium on High Performance Distributed Computing*, pages 23–30, Washington, DC, August 1995.

[158] J. C. Tannehill, D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Series in Computational and Physical Processes in Fluid

Mechanics and Thermal Sciences. Hemisphere Pub. Corp., Taylor & Francis, Washington, DC, 2nd edition, 1997.

[159] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction.* Springer-Verlag, New York, 2nd edition, 1999.

[160] S. Tu. *Development of a Solution Adaptive Cartesian-Grid Solver for 2-D Thermochemical Nonequilibrium Flows.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, November 2001.

[161] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna. A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries. *Journal of Computational Physics*, 174(1):345–380, 2001.

[162] W. R. Van Dalsem and J. L. Steger. Finite-Difference Simulation of Transonic Separated Flow Using a Full Potential-Boundary Layer Interaction Approach. In *AIAA 16th Fluid and Plasma Dynamics Conference*, Danvers, MA, July 1983. AIAA-83-1689.

[163] M. Van Dyke. Higher-Order Boundary-Layer Theory. In W. R. Sears and M. Van Dyke, editors, *Annual Review of Fluid Mechanics*, volume 1, pages 265–292. Annual Reviews, Inc., Palo Alto, CA, 1969.

[164] B. van Leer. Towards the Ultimate Conservative Difference Scheme III: Upstream-Centered Finite Difference Schemes for Ideal Compressible Flow. *Journal of Computational Physics*, 23:263–275, 1977.

[165] B. van Leer. Towards the Ultimate Conservative Difference Scheme IV: A New Approach to Numerical Convection. *Journal of Computational Physics*, 23:276–299, 1977.

[166] B. van Leer. Towards the Ultimate Conservative Difference Scheme V: A Second Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32:101–136, 1979.

[167] B. van Leer, C. H. Tai, and K. G. Powell. Design of Optimally-Smoothing Multi-Stage Schemes for the Euler Equations. In *9th AIAA Computational Fluid Dynamics Conference*, Washington, DC, July 1989. AIAA-89-1933-CP.

[168] V. N. Vatsa and J. E. Carter. Analysis of Airfoil Leading Edge Separation Bubbles. In *AIAA 21st Aerospace Sciences Meeting*, Reno, NV, January 1983. AIAA-83-0300.

[169] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. In *AIAA 31st Aerospace Sciences Meeting*, Reno, NV, January 1993. AIAA-93-0880.

[170] V. Venkatakrishnan. Parallel Implicit Unstructured Grid Euler Solvers. ICASE Report 94-04, ICASE, Hampton, VA, January 1994. NASA/CR-191594.

[171] V. Venkatakrishnan. Implicit Schemes and Parallel Computing in Unstructured Grid CFD. ICASE Report 95-28, ICASE, Hampton, VA, April 1995. NASA/CR-195071.

[172] V. Venkatakrishnan. A Perspective on Unstructured Grid Flow Solvers. ICASE Report 95-03, ICASE, Hampton, VA, January 1995. NASA/CR-195025.

[173] V. Venkatakrishnan and H. D. Simon. A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids. *The Journal of Supercomputing*, 6(2):117–137, June 1992.

[174] M. Vinokur. Conservation Equations of Gasdynamics in Curvilinear Coordinate Systems. *Journal of Computational Physics*, 14(1):105–125, 1974.

[175] R. G. Voigt. Where are the Parallel Algorithms? ICASE Report 85-02, ICASE, Hampton, VA, January 1985. NASA/CR-172516.

[176] D. Voorhies. Space-Filling Curves and a Measure of Coherence. In J. Arvo, editor, *Graphic Gems II*, The Graphic Gem Series, pages 26–30. Academic Press, Inc., New York, NY, 1991.

[177] R. W. Walters and J. L. Thomas. Advances in upwind relaxation methods. In A. K. Noor and J. T. Oden, editors, *State-of-the-Art Surveys on Computational Mechanics*, pages 145–183. The American Society of Mechanical Engineers, 1989.

[178] G. Wang, L. N. Sankar, and H. Tadghaghi. Prediction of Rotorcraft Noise with a Low-Dispersion Finite Volume Scheme. *AIAA Journal*, 38(3):395–401, March 2000.

[179] P. Wang. Massively Parallel Finite Volume Computation of Three-Dimensional Thermal Convective Flows. *Advances in Engineering Software*, 29(3–6):307–315, 1998.

[180] Z. J. Wang. A Fast Nested Multi-Grid Viscous Flow Solver for Adaptive Cartesian/Quad Grids. In *27th AIAA Fluid Dynamics Conference*, New Orleans, LA, June 1996. AIAA-96-2091.

[181] Z. J. Wang. A Global BMRES/Multi-Grid Scheme for an Adaptive Cartesian/Quad Grid Flow Solver On Distributed Memory Machines. In *13th AIAA Computational Fluid Dynamics Conference*, Snowmass Village, CO, June 1997. AIAA-96-2091.

[182] Z. J. Wang. A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equations. *Computers & Fluids*, 27(4):529–549, 1998.

[183] Z. J. Wang and Y. Sun. A Curvature-Based Wall Boundary Condition for the Euler Equations on Unstructured Grids. In *40th Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2002. AIAA. AIAA-2002-0966.

[184] B. Wedan and J. C. South, Jr. A Method for Solving the Transonic Full-Potential Equation for General Configurations. In *AIAA 6th Computational Fluid Dynamics Conference*, Danvers, MA, July 1983. AIAA-83-1889.

[185] Y. Wenren, M. Fan, W. Dietz, G. Hu, C. Braun, and J. Steinhoff. Efficient Eulerian Computation of Realistic Rotorcraft Flows Using Vorticity Confinement - A Survey of Recent Results. In *AIAA 39th Aerospace Sciences Meeting*, Reno, NV, January 2001. AIAA-2001-0996.

[186] F. M. White. *Viscous Fluid Flow*. McGraw-Hill Series in Mechanical Engineering. McGraw-Hill, New York, 2nd edition, 1991.

[187] Z.-N. Wu and H. Zou. Grid Overlapping for Implicit Parallel Computation of Compressible Flows. *Journal of Computational Physics*, 157(1):2–43, 2000.

[188] G. Yang, D. M. Causon, D. M. Ingram, R. Saunders, and P. Batten. A Cartesian Cut Cell Method for Compressible Flows Part A: Static Body Problems. *The Aeronautical Journal*, 101(2):47–56, February 1997.

[189] G. Yang, D. M. Causon, D. M. Ingram, R. Saunders, and P. Batten. A Cartesian Cut Cell Method for Compressible Flows Part B: Moving Body Problems. *The Aeronautical Journal*, 101(2):57–65, February 1997.

[190] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy. An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries. *Journal of Computational Physics*, 156(2):209–240, 1999.

[191] T. Ye, R. R. Mittal, H. S. Udaykumar, and W. Shyy. A Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries. In *AIAA 3rd Weakly Ionized Gases Workshop*, Norfolk, VA, November 1999. AIAA. AIAA-99-3312.

[192] H. Y. Yoon, S. Koshizuka, and Y. Oka. Particle-Gridless Hybrid Method for Incompressible Flows. *International Journal for Numerical Methods in Fluids*, 30(4):407–424, 1999.

[193] H. Youngren and M. Drela. Viscous/Inviscid Method for Preliminary Pesign of Transonic Cascades. In *27th AIAA, SAE, ASME, and ASEE, Joint Propulsion Conference*, Sacramento, CA, June 1991. AIAA-91-2364.

[194] H. Zhang, M. Reggio, J. Y. Trepanier, and R. Camarero. Discrete Form of the GCL for Moving Meshes and Its Implementation in CFD Schemes. *Computers & Fluids*, 22(1):9–23, 1993.

VITA

David D. Marshall was born in Tonawanda, New York, USA on March 20, 1972. He received his B.S. degree in Mechanical Engineering with Aerospace Interests from Worcester Polytechnic Institute, Worcester, Massachusetts, USA in February 1994. He then entered the School of Aerospace Engineering at Georgia Institute of Technology in Atlanta, Georgia, USA and received his M.S. degree in Aerospace Engineering in August of 1995. After leaving Georgia Tech, he spent four years working, first at Lockheed-Martin Management & Data Systems in Springfield, Virginia, USA for one year and then at Avtec Systems in Fairfax, Virginia until returning to Georgia Tech in September 1999 to enter the Aerospace Engineering doctoral program.